

Università degli Studi di Camerino

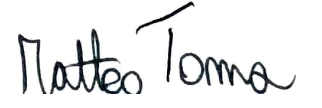
Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica (Classe L-31)

Archiviazione dello stato della sessione

di un’applicazione ASP.NET con Redis vs SQL Server

Laureando Relatore

Matteo Toma Prof. Fausto MarcantoniImmagine che contiene testo

Descrizione generata automaticamente

Matricola 116781

Correlatore

Prof. Roberto Gagliardi

A.A. 2021/2022

# Indice

[Indice 2](#_Toc115799429)

[1. Abstract 3](#_Toc115799430)

[1.1 Motivazione 3](#_Toc115799431)

[1.2 Obiettivi 3](#_Toc115799432)

[1.3 Struttura della Tesi 4](#_Toc115799433)

[2. Introduzione 5](#_Toc115799434)

[2.1 Cos’è ASP.NET 5](#_Toc115799435)

[2.2 Cos’è Redis 7](#_Toc115799436)

[3. Memorizzazione dello stato della sessione 11](#_Toc115799437)

[3.1 Introduzione 11](#_Toc115799438)

[3.2 Creazione di una Server Farm Locale 14](#_Toc115799439)

[3.3 Definizione delle regole di riscrittura degli URL 19](#_Toc115799440)

[4. Implementazione dell’applicazione in ASP.NET 24](#_Toc115799441)

[4.1 Scrittura del codice 24](#_Toc115799442)

[4.2 Pubblicazione ed uso dell'applicazione 30](#_Toc115799443)

[5. Memorizzazione dei dati e dello stato della sessione 34](#_Toc115799444)

[5.1 Installazione e avvio di Redis 34](#_Toc115799445)

[5.2 Memorizzazione dello stato della sessione su Redis 35](#_Toc115799446)

[5.3 Memorizzazione dello stato della sessione su Sql Server 38](#_Toc115799447)

[6. Analisi delle prestazioni 43](#_Toc115799448)

[6.1 Test delle prestazioni con SQL SERVER 43](#_Toc115799449)

[6.2 Test delle prestazioni con REDIS 50](#_Toc115799450)

[Conclusioni e Sviluppi Futuri 67](#_Toc115799451)

[Bibliografia 68](#_Toc115799452)

[Elenco delle figure 70](#_Toc115799453)

[Elenco dei grafici 72](#_Toc115799454)

[Ringraziamenti 73](#_Toc115799455)

# Abstract

Questa tesi vuole mostrare come salvare lo stato della sessione di un’applicazione creata con ASP.NET attraverso l’uso di un database open-source chiamato Redis.

Dopo aver definito l’applicazione da realizzare, si andrà a definire lo schema concettuale di dati da dover memorizzare nel database, successivamente si analizzeranno delle tecniche per poter memorizzare lo stato della sessione di un’applicazione.

Infine, si salveranno i dati in due diversi database, SQL server e Redis, andando a fare delle analisi di performance su di essi.

## **Motivazione**

Il motivo per cui si è scelto di usare queste due tecnologie proviene dalla necessità di progettare un’applicazione in modo semplice ma efficace. Grazie alla semplicità di uso del framework ASP.NET, Redis e Sql Server, raggiungere gli obiettivi prefissati sarà più chiaro e veloce.

## **Obiettivi**

Gli obiettivi della tesi sono:

* Creare l’applicazione su ASP.NET
* Memorizzare i dati dell’applicazione
* Memorizzare lo stato della sessione dell’applicazione
* Dimostrare quale tra SQL Server e Redis è più performante.

## **Struttura della Tesi**

La struttura della Tesi è descritta di seguito:

1. Introduzione ad ASP.net
2. Introduzione a Redis
3. Introduzione a SQL Server
4. Creazione dell’applicazione
5. Memorizzazione dei dati
6. Analisi di performance sui dati memorizzati

Nella prima, seconda e terza sezione si andrà a descrivere cosa sono e a cosa servono ASP.net, Redis e SQL Server.

Nella quarta come l’applicazione è implementata, con vari casi illustrativi.

Nella quinta sezione si descriverà come e dove saranno memorizzati tutti i dati riguardanti l’applicazione e come memorizzare lo stato della sessione.

Nell’ultima sezione si farà un’analisi sulle performance dei due diversi database utilizzati, per capire quale sia il migliore con applicazioni sviluppate in ASP.net.

# Introduzione

# Cos’è ASP.NET

*ASP.NET* è un framework nato 2002.  
Dalla sua nascita si è puntato, sempre di più, a programmare le applicazioni web nel modo più semplice possibile.

Contemporaneamente, era necessario memorizzare tutti i dati di queste applicazioni in un server sicuro ed efficace. Ce ne sono molti, ma ognuno ha dei pro e contro, offre prestazioni diverse e ha diversi scopi specifici per ogni applicazione in base alle necessità.

ASP.NET è un framework Web gratuito, commercializzato da Microsoft , per la creazione di siti Web e applicazioni Web usando HTML, CSS e JavaScript. "È anche possibile creare API Web e usare tecnologie in tempo reale come Web Socket." (“panoramica ASP.NET | Microsoft Learn”)

"ASP.NET offre tre framework per la creazione di applicazioni Web:" (“panoramica ASP.NET | Microsoft Learn”)

1. Web Forms
2. ASP.NET MVC
3. Pagine Web ASP.NET.

Tutti e tre i framework sono ormai stabili ed è possibile creare applicazioni Web con qualità elevata con qualunque di essi.

Con i Web Form, si può ottenere uno sviluppo rapido usando una libreria avanzata di controlli che incapsulano il markup HTML.

É, inoltre, possibile creare dei siti Web dinamici usando un modello basato su trascinamento e rilascio, molto più facile da capire. Questo modello di programmazione offre agli sviluppatori un'area di progettazione con numerosi controlli e componenti per creare rapidamente siti Web basati su interfaccia utente molto efficaci. (“Creare il | Microsoft Learn”)

Attraverso ASP.NET MVC, invece, si ha il controllo completo sul markup HTML, il codice e il markup separati e i test di scrittura.

ASP.NET MVC offre opzioni avanzate , basate su criteri per creare siti Web dinamici che consentono di ottenere una netta separazione degli aspetti problematici e garantiscono il controllo completo del markup per permettere uno sviluppo semplice e flessibile. (“panoramica ASP.NET | Microsoft Learn”)

Attraverso le pagine Web, infine, si può ottenere Markup HTML e codice insieme nello stesso file.

Le pagine Web ASP.NET e la sintassi Razor (un sistema di scripting lato server)

offrono un modo rapido e leggero per combinare il codice server con HTML per creare contenuto Web dinamico. (“panoramica ASP.NET | Microsoft Learn”)

Connettersi ai database, aggiungere video, collegare ai siti di social networking e includere molte altre funzionalità che consentono di creare siti conformi agli standard Web più recenti. (“panoramica ASP.NET | Microsoft Learn”)

**Vantaggi** dell’uso di ASP.NET:

1. **Prestazioni**: è più veloce degli altri framework web disponibili sul mercato.

Codice di backend: con l'aiuto di ASP.NET è possibile scrivere il codice per il backend per l'accesso ai dati e qualsiasi logica in C#.

1. **Pagine dinamiche**: in ASP.NET, Razor fornisce la sintassi per lo sviluppo di pagine Web dinamiche con l'aiuto di C# e HTML. "ASP.NET può essere integrato con JS (JavaScript) e include anche i framework come React e Angular per la SPA (Single Page Application.)" (“Le 50 migliori domande e risposte per le interviste ASP.NET”)
2. **Supporto di diversi sistemi operativi**: è possibile sviluppare ed eseguire app ASP.NET su Windows, Linux, Docker e MacOS. Inoltre, Visual Studio fornisce gli strumenti necessari per creare app .NET con diversi sistemi operativi.

## **Cos’è Redis**

**"Redis**, acronimo di Remote Dictionary Server, è un archivio dati veloce, open source, in memoria e di tipo chiave-valore." (“Redis: datastore in memoria. Come funziona e perché devi utilizzarlo”) Il progetto fu avviato quando Salvatore Sanfilippo, lo sviluppatore originale di Redis, ha voluto migliorare la scalabilità della sua startup italiana. Da lì, ha sviluppato Redis, che ora viene utilizzato come database, cache e broker di messaggi e coda.

Si tratta di un database non relazionale o No Sql.

Redis offre tempi di risposta inferiori al millisecondo, consentendo milioni di richieste al secondo per applicazioni in tempo reale in settori come videogiochi, tecnologia pubblicitaria, servizi finanziari, sanità e IoT. (“Redis: datastore in memoria. Come funziona e perché devi utilizzarlo”) Ad oggi, Redis è uno dei database open source più popolari, definito il “più amato" da Stack Overflow (un importante sito Q&A usato da milioni di programmatori in tutto il mondo) per cinque anni consecutivi.

Figura

Immagine che contiene testo, parete, monitor, nero

Descrizione generata automaticamente

**Vantaggi di Redis**

**Prestazioni**

Tutti i dati Redis risiedono in memoria, il che abilita l'accesso ai dati a bassa latenza e velocità effettiva elevata. A differenza dei database tradizionali, gli archivi dati in memoria non richiedono un viaggio su disco, riducendo la latenza del motore a dei microsecondi. Per questo motivo, gli archivi dati in memoria possono supportare una quantità di operazioni di ordine di grandezza superiore e tempi di risposta più rapidi. (“Redis: datastore in memoria. Come funziona e perché devi utilizzarlo”)

Si ottengono quindi prestazioni molto veloci con operazioni di lettura e scrittura medie che richiedono meno di un millisecondo e supporto per milioni di operazioni al secondo. (“Redis: datastore in memoria. Come funziona e perché devi utilizzarlo”) Nei prossimi capitoli si andranno a testare.

**Strutture dati flessibili**

A differenza di altri data store chiave-valore che offrono strutture dati limitate, Redis possiede un'ampia molteplicità di strutture dati per soddisfare le esigenze dell’applicazione. I tipi di dati Redis includono:

* "String (stringhe): testo o dati binari di dimensioni fino a 512 MB" (“Redis: datastore in memoria. Come funziona e perché devi utilizzarlo”)
* List (elenchi): una raccolta di stringhe
* "Set: una raccolta non ordinata di stringhe con capacità di intersezione, unione e differenza di altri tipi set" (“Redis: datastore in memoria. Come funziona e perché devi utilizzarlo”)
* Sorted Set (set memorizzati): set ordinati per un valore
* Hash: strutture dati per archiviare elenchi di campi e valori
* Bitmap: un tipo di dato che offre operazioni sui bit
* HyperLog: una struttura di dati probabilistica per la stima di voci uniche in un dataset
* Streams: I flussi sono una struttura di dati di sola aggiunta (si possono usare per tracciare le azioni dell’utente, i clicks, o memorizzare le notifiche in uno stream separato e monitorare sensori da diversi dispositivi.)
* Geo spazialità: mappe di voci basate su longitudine/latitudine, "vicinanza"
* JSON: un oggetto semi-strutturato nidificato formato da valori denominati. Supporta numeri, stringhe, valori booleani, array e numerosi altri oggetti.

**Semplice e intuitivo**

Redis abilita a scrivere codice tradizionalmente complesso ma con meno righe più semplici.

Si possono scrivere meno righe di codice per archiviare, accedere e utilizzare i dati nelle tue applicazioni. La differenza è questa: gli sviluppatori che usano Redis possono utilizzare una semplice struttura di comandi rispetto ai linguaggi di query dei database tradizionali come SQL Server, MySql o PostgreSQL.

Ad esempio, per utilizzare la struttura dati hash Redis per spostare i dati in un archivio dati basta una sola riga di codice.

"Per gli sviluppatori Redis sono disponibili più di cento client open source." (“Redis: datastore in memoria. Come funziona e perché devi utilizzarlo”)

I linguaggi supportati sono moltissimi:

ActionScript, C, C++, C#, Chicken, Clojure, Common Lisp, Crystal, D, Dart, Elixir, Erlang, Go, Haskell, Haxe, Io, Java, Nim, JavaScript (Node.js), Julia, Lua, Objective-C, OCaml, Perl, PHP, Pure Data, Python, R, Racket, Ruby, Rust, Scala, Smalltalk, Swift.

**Replica e persistenza**

Utilizza un'architettura primary-replica (un modello di comunicazione in cui un dispositivo ha il controllo unidirezionale su uno o più altri dispositivi ).

Supporta inoltre la replica asincrona, cioè i dati vengono replicati su diversi server appositi. In questo modo è possibile ottenere migliori prestazioni in lettura (poiché le richieste vengono suddivise tra i diversi server) e il ripristino in caso di interruzione del server principale. (“Cos'è Redis? – Amazon Web Services (AWS)”)

Riguardo la persistenza dei dati, Redis supporta dei backup point-in-time (copia su disco dell’intero dataset Redis).

**Disponibilità e scalabilità elevate**

Redis utilizza l’architettura primary-replica in un singolo nodo principale o una topologia a cluster. "In questo modo è possibile creare soluzioni dotate di elevata disponibilità e prestazioni costanti e affidabili." (“Redis: datastore in memoria. Come funziona e perché devi utilizzarlo”)

I cluster servono a partizionare automaticamente i dati su più nodi Redis.

Le dimensioni di un cluster possono essere modificate in vari modi, secondo modelli di scalabilità orizzontale e verticale.

**Open source**

Redis è un progetto open source supportato da un’attiva community, tra cui AWS (Amazon Web Services). Non esistono vincoli di fornitore o di tecnologia: è basato su standard aperti, supporta formati di dati aperti e funziona con diversi client. (“Redis: datastore in memoria. Come funziona e perché devi utilizzarlo”)

Le compatibilità con i sistemi operativi sono illustrate nella seguente figura:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

# Memorizzazione dello stato della sessione

## **Introduzione**

Nel caso di applicazioni web importanti, molte applicazioni spesso sono ospitate su più di un server. Nei casi in cui si voglia avere un backup di un server nel caso in cui uno si guasti, o si voglia bilanciare il carico delle richieste su più server, si può utilizzare una server farm locale.

Per poter memorizzare lo stato di un’archiviazione della sessione dell’applicazione che si sta utilizzando (ad esempio, nel caso di un sito web e-commerce si vuole memorizzare gli articoli aggiunti al carrello grazie alle sessioni), usare una server farm locale è l’ideale.

Ma, poiché vi sono più server, se qualcosa andasse storto la memorizzazione delle sessioni potrebbe andare persa (se, ad esempio, viene chiusa la finestra del browser per sbaglio).

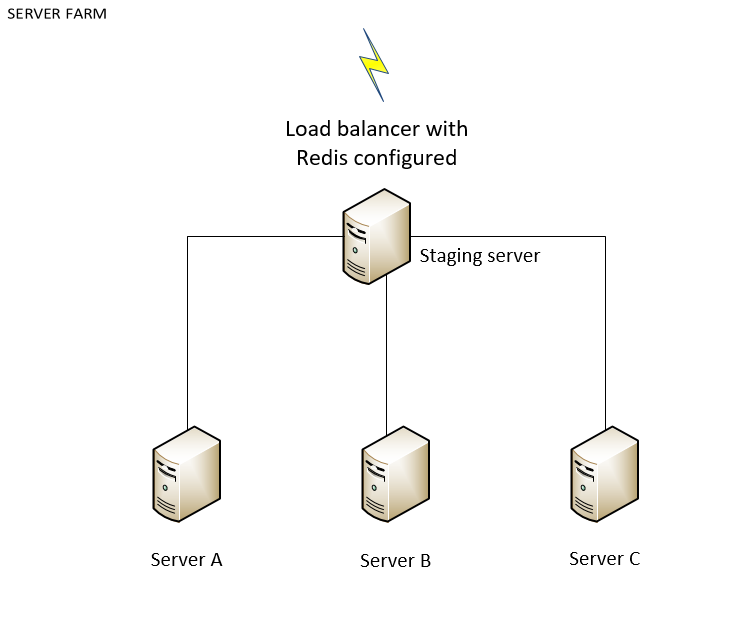
Grazie all’utilizzo di Redis, essendo tale un archivio dati in memoria, si trova sul server di bilanciamento del carico.

Le richieste arrivano al sistema di bilanciamento del carico nel caso in cui Redis viene usato per archiviare le sessioni.

In questo caso, quando una richiesta successiva arriva nelle sessioni precedenti può essere trovata nel bilanciamento del carico grazie a Redis.

L’autore dell’articolo Bibliografia[Fiy2015] mostra un esempio in cui si hanno quattro server, come in figura:

Figura



Quindi nella server farm, ad esempio, possono esserci quattro server, tra cui uno server per bilanciare il carico delle richieste. Viene chiamato “staging server”. Tutte le richieste che arrivano al server di staging vengono bypassate su uno dei tre server collegati, in cui si trovano le istanze effettive dell’applicazione web.

## **Creazione di una Server Farm Locale**

Per creare la Server Farm in Locale, servirà IIS.

Internet Information Services (IIS) è un server Web flessibile e generico di Microsoft che viene eseguito su sistemi Windows per servire le pagine o i file HTML richiesti. (“Velocità dei Dati e Download di file da Server a Client”)

Accetta le richieste dai computer client remoti e restituisce la risposta appropriata. (“Velocità dei Dati e Download di file da Server a Client”) Questa funzionalità consente ai server Web di condividere e fornire informazioni su reti locali (LAN), come intranet aziendali e WAN (Wide Area Network), come Internet.

Può fornire le informazioni agli utenti in diverse forme, ad esempio pagine Web statiche in HTML; scambi di file come download e upload, e documenti di testo, file di immagine e altro ancora.

IIS Express invece è una versione leggera e autonoma di IIS ottimizzata per gli sviluppatori. (“Domande frequenti su IIS Express | Microsoft Learn”) Ha tutte le funzionalità di base di IIS 7 e versioni successive, e funzionalità aggiuntive quali:

1. Non viene eseguito come servizio né richiede diritti utente amministratore per eseguire la maggior parte delle attività.
2. Funziona bene con le applicazioni ASP.NET e PHP.
3. Più utenti di IIS Express possono lavorare in modo indipendente sullo stesso computer.

Per abilitare IIS, si apre il pannello di controllo su Windows, successivamente programmi e Funzionalità e Attivazione o disattivazione delle funzionalità di Windows, e selezionare, come in figura, Estendibilità .NET 4.8:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

e Strumenti di Gestione Web:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Basterà quindi riavviare il Computer e digitare nella barra di ricerca IIS.

Per poter creare una server farm da IIS manager sono necessarie diverse estensioni, una di queste è chiamata Application Request Routing (ARR) per IIS, installabile dal sito della Microsoft:

Server Web Platform:

<https://www.microsoft.com/web/downloads/platform.aspx>

E l’ARR al seguente link:

<https://www.microsoft.com/en-us/download/details.aspx?id=47333>

Infine, l’estensione Web Deploy 3.6:

<https://www.microsoft.com/it-it/download/confirmation.aspx?id=43717>

Una volta installate queste estensioni, bisogna creare dei server locali per la server farm.

Visto che si stanno utilizzando delle porte diverse su localhost per poter agire tra più server, bisogna includere l’indirizzo di questi server nel file hosts di Windows, (presente nella cartella C:\Windows\System32\drivers\etc\hosts).

Bisogna assegnare tre indirizzi IP per i tre server (server staging, server a, server b), come in figura:

Figura

Immagine che contiene testo

Descrizione generata automaticamentePer distinguere i tre server, bisogna vincolare su di loro le porte.

Per farlo, basta aprire IIS, e col click destro nella sezione Siti, selezionare la voce “Aggiungi sito Web”.

Aggiunti i tre siti per i server web, ovvero bilanciamento del carico, server-a e server-b, si andranno a configurare le porte dei server: la porta 9000 per il bilanciamento del carico, 9001 per il server-a e 9002 è per il server-b.

In figura un esempio di configurazione:

Figura

Immagine che contiene testo

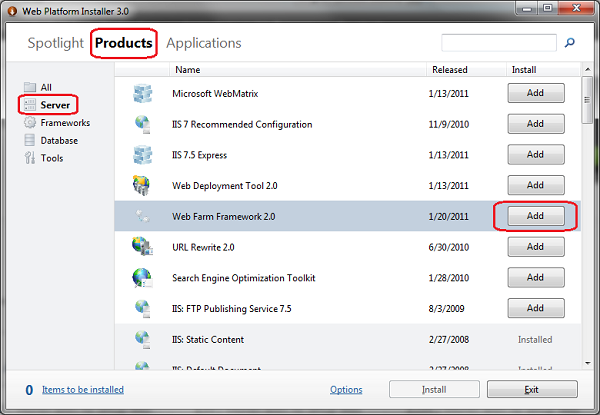
Descrizione generata automaticamente

N.B.: Il default Web Site è il sito di default presente nella path di Windows in figura, dopo aver abilitato IIS.

Ora è necessario aggiungere la server Farm, per farlo basterà aprire Web Platform Installer e tra i prodotti aggiungere Web Farm FrameWork.

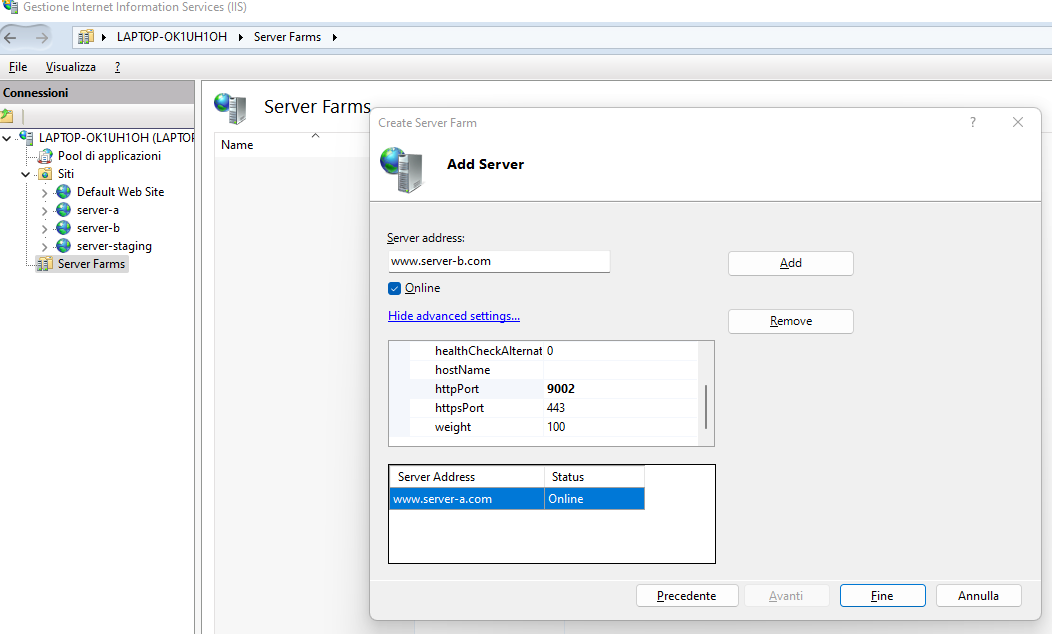
Nella figura sottostante è illustrato come fare:

Figura



Su IIS, si crea una Server Farm con il click destro del mouse su “Server Farms” e si aggiungono i due server a e b, con le relative porte 9000 e 9001, come in figura sottostante:

Figura



## **Definizione delle regole di riscrittura degli URL**

Il modulo di riscrittura degli URL consente agli amministratori di IIS di creare potenti regole personalizzate per mappare gli URL delle richieste a URL descrittivi. Essi sono più facili da ricordare per gli utenti e più facili da trovare per i motori di ricerca.   
Inoltre:

* + - Si possono inoltre sostituire facilmente gli URL delle applicazioni Web per produrre risultati intuitivi sia per l'utente che per i motori di ricerca.
    - "Riscrivere gli URL in base alle intestazioni HTTP e alle variabili del server IIS." (“Uso del modulo di riscrittura URL | Microsoft Learn”)
    - Si possono eseguire reindirizzamenti, inviare risposte personalizzate o interrompere le richieste HTTP in base alla logica definita nelle regole di riscrittura. (“IIS installazione URL Rewrite e reindirizzamente HTTTP ad HTTPS”)
    - Infine, è possibile controllare l’accesso al contenuto dei siti Web in base ai segmenti URL o ai metadati della richiesta.

Il modulo è scaricabile al link seguente:

<https://www.microsoft.com/web/handlers/webpi.ashx?command=getinstallerredirect&appid=urlrewrite2>

Dopodiché, si seleziona Url Rewrite, come in figura:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Poi, con il click destro del mouse, si aggiunge una regola:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Si seleziona il tipo di regola (blank rule) e s’imposta un nome (in questo caso, ad esempio, redis-farm-rule).

Poi si aggiungono due condizioni per il reindirizzamento dei link:

Figura

Immagine che contiene tavolo

Descrizione generata automaticamente

Successivamente, sulla sezione Azione s’imposta “Instrada a server farm”, selezionando “interrompi elaborazione regole successive”:

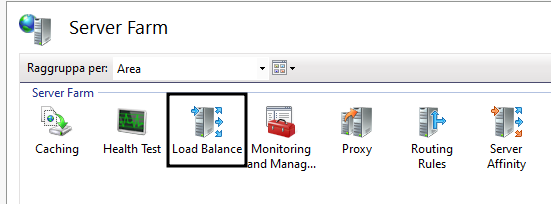
Figura

Immagine che contiene testo

Descrizione generata automaticamente

Dopo aver cliccato su applica, bisogna configurare l’algoritmo di bilanciamento delle richieste del Server.  
Per farlo, bisogna aprire la server farm e selezionare “Load Balance”:

Figura



e selezionare Round Robin:

Figura

Immagine che contiene testo

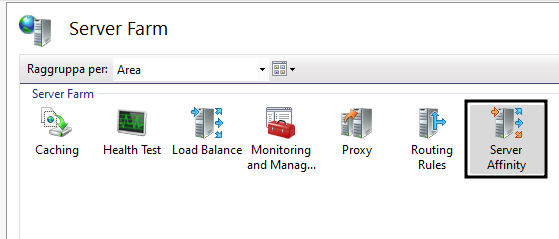
Descrizione generata automaticamente

Questo algoritmo è uno dei più semplici per distribuire le richieste dei client su un gruppo di server. Scendendo nell'elenco dei server nel gruppo, l’algoritmo inoltra una richiesta client a ciascun server a turno. Quando raggiunge la fine dell'elenco, il sistema di bilanciamento del carico torna indietro e riscende di nuovo nell'elenco (invia la richiesta successiva al primo server elencato, quella successiva al secondo server e così via).

Quindi nel caso in questione, se una richiesta arriva per la prima volta al server di bilanciamento del carico, verrà instradata al server A. Se poi la pagina viene ricaricata e la richiesta viene inviata di nuovo, verrà instradata al server B. I server verranno ripristinati ogni volta che arriva una richiesta.

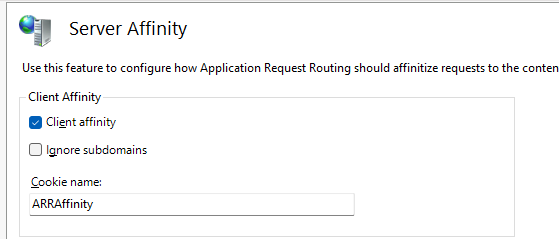
Ora bisogna impostare l’affinità dei Server, per farlo, ritornare alla pagina della server farm e selezionare “Server Affinity”:

Figura



Infine, selezionare l’opzione “Client Affinity”:

Figura



Nel successivo capitolo, si passerà passare alla creazione l’applicazione in ASP.NET WEB per testare la Server Farm.

# Implementazione dell’applicazione in ASP.NET

In questo capitolo si andrà a scrivere il codice dell’applicazione in ASP.NET da pubblicare sul server web della farm locale.

Si creerà un semplice form per acquisire dati dal server web, per poi memorizzarli sia su un server SQL sia su REDIS.

## **Scrittura del codice**

Per creare un nuovo progetto, bisogna aprire visual studio e selezionare la categoria Web e Cloud e tra le opzioni, “Sviluppo ASP.NET e Web”

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Poi si seleziona il tipo di progetto:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

E s’inserisce il nome dell’applicazione:

Figura

Immagine che contiene testo, screenshot, monitor, schermo

Descrizione generata automaticamente

Si aggiunge un Web Form:

Figura

Immagine che contiene testo, monitor, screenshot, schermo

Descrizione generata automaticamente

Verrà creato il file WebForm1.aspx, che si andrà a modificare per personalizzare il form.

In questo caso il form viene chiamato Index.aspx per rappresentare la pagina iniziale.

Il secondo file è il Result.aspx, che rappresenta la pagina di Redirect dopo aver premuto il bottone “Presentati” nella pagina Index.aspx.

La struttura del progetto, contenente file e cartelle, è la seguente:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Si può proseguire con la scrittura del codice del file Index.aspx. Il contenuto è il seguente:

*<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Index.aspx.cs" Inherits="SqlServerVsRedis.\_Index" %>*

*<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Index.aspx.cs" Inherits="SqlServerVsRedis.\_Index" %>*

*<!DOCTYPE html>*

*<html xmlns="http://www.w3.org/1999/xhtml">*

*<head runat="server">*

*<title></title>*

*</head>*

*<body>*

*<form id="form1" runat="server">*

*<div>*

*<h1>Memorizzare i Dati di una Sessione in Azure Redis Cache</h1>*

*<asp:TextBox ID="NameTextBox" runat="server"></asp:TextBox>*

*<asp:Button ID="GreetButton" runat="server" Text="Presentarsi" OnClick="GreetButton\_OnClick" />*

*</div>*

*</form>*

*</body>*

*</html>*

Mentre il contenuto del file Index.aspx.cs è questo:

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Web;*

*using System.Web.UI;*

*using System.Web.UI.WebControls;*

*namespace SqlServerVsRedis*

*{*

*public partial class \_Index : System.Web.UI.Page*

*{*

*protected void Page\_Load(object sender, EventArgs e)*

*{*

*}*

*protected void GreetButton\_OnClick(object sender, EventArgs e)*

*{*

*string name = NameTextBox.Text;*

*try*

*{*

*Session["user"] = NameTextBox.Text;*

*Session["time"] = DateTime.Now;*

*Response.Redirect("~/Result.aspx");*

*}*

*catch (Exception ex)*

*{*

*NameTextBox.Text = ex.Message;*

*}*

*}*

*}*

*}*

Poi si aggiunge un file .aspx con nome Result.aspx, che conterrà il risultato della richiesta al server.

Il codice è il seguente:

*<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Result.aspx.cs" Inherits="SqlServerVsRedis.Result" %>*

*<!DOCTYPE html>*

*<html xmlns="http://www.w3.org/1999/xhtml">*

*<head runat="server">*

*<title></title>*

*</head>*

*<body>*

*<form id="form1" runat="server">*

*<div>*

*<asp:Label ID="NameLabel" runat="server" Text="Label"></asp:Label>*

*</div>*

*</form>*

*</body>*

*</html>*

Mentre il contenuto del file Result.aspx.cs è questo:

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Web;*

*using System.Web.Services.Description;*

*using System.Web.UI;*

*using System.Web.UI.WebControls;*

*using System.Xml;*

*using System.Xml.Linq;*

*using Microsoft.AspNet.SessionState;*

*using StackExchange.Redis;*

*namespace SqlServerVsRedis {*

*public partial class Result : System.Web.UI.Page {*

*protected void Page\_Load(object sender, EventArgs e) {*

*NameLabel.Text = "Ciao " + (string)Session["user"] + ", l'ultimo tempo di sessione memorizzata è stato: " + Session["time"];*

*}*

*}*

*}*

## **Pubblicazione ed uso dell'applicazione**

Per pubblicare il contenuto delle pagine sul server, bisogna avviare visual studio come amministratore (per poter concedere a Visual Studio i permessi di scrittura e lettura sul disco) e con il tasto destro sul progetto, bisogna cliccare su pubblica, come in figura:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Successivamente, si seleziona la destinazione della pubblicazione, in questo caso in locale:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Dopodiché, s’inserisce la path di destinazione:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Premendo Ok, verrà mostrata la seguente schermata:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Dopo aver premuto il pulsante pubblica su Visual Studio, verranno caricati i file nella cartella scelta, come in figura:

Figura

Immagine che contiene testo, monitor, nero, schermo

Descrizione generata automaticamente

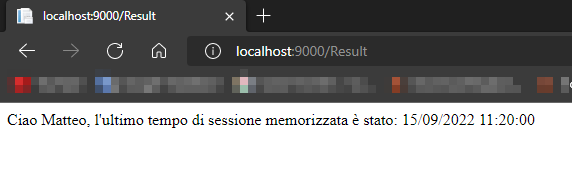
Aprendo ora un Web Browser a scelta (nel caso in questione Microsoft Edge), e andando all’indirizzo configurato nel **Capitolo 3 sezione 3.3,** cioè localhost:9000/Result.aspx, verrà mostrata la seguente pagina:

Figura



Una volta inserito il nome e premuto il bottone “Presentati”, si verrà indirizzati alla pagina Result.aspx, che conterrà il nome dato in input nella pagina Index, e il tempo in cui la sessione memorizzata è stata creata:

Figura



# Memorizzazione dei dati e dello stato della sessione

## **Installazione e avvio di Redis**

Per l’installazione di Redis su Windows si può utilizzare il pacchetto “Nuget” e “Chocolatey”, nel caso in questione si è scelto il secondo.

Prima bisogna installare Chocolatey (un gestore di pacchetti e installatore per Windows), aprendo il cmd e digitando;

*@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "[System.Net.ServicePointManager]::SecurityProtocol = 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"*

Successivamente, sempre nel prompt dei comandi, bisogna digitare

*choco install redis-64 --version=3.0.501*

Una volta installato, i componenti saranno nella cartella *C:\ProgramData\chocolatey\lib\redis-64*

Infine, per avviare redis, bisogna aprire redis-cli:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

## **Memorizzazione dello stato della sessione su Redis**

Per completare il processo di memorizzazione, bisogna installare un pacchetto nuget che aiuti a memorizzare gli stati della sessione in Redis.

Il pacchetto si chiama RedisSessionStateProvider.

Per installarlo basterà digitare il comando

*Install-Package Microsoft.Web.RedisSessionStateProvider*

Nella console della gestione dei pacchetti di Visual Studio

E per fare in modo che Redis memorizzi le nostre sessioni, dobbiamo aggiungere queste righe al nostro file di configurazione web.

Per aprire la finestra di comando, basterà fare come in figura:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Una volta installato il pacchetto, per memorizzare gli stati della sessione, è necessario aprire il file Web.config e aggiungere l’host nei provider, e modificando l’SSL da true e false (per gli scopi del caso in studio non è necessario), come in figura:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Una volta inserito il nome e aver premuto sulla pagina Index, il bottone “Presentati”, verranno memorizzati sul server nella cache di Redis i dati della sessione, che saranno criptati:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Se si vuole eliminare una delle chiavi, si digita il comando DEL e il nome della chiave da cancellare, ad esempio,

*DEL KEYS "{/\_p51qcltln34mnstwyh5flpow}\_Data"* ,

e la prima chiave verrà eliminata.

Se invece si vuole eliminare tutto, basta scrivere *flushall* .

Nella sessione successiva si tratterà la memorizzazione dello stato della sessione su Sql Server.

## **Memorizzazione dello stato della sessione su Sql Server**

Per memorizzare lo stato della sessione su Sql Server, per prima cosa è necessario installarlo.

È possibile farlo al seguente link: <https://info.microsoft.com/ww-landing-sql-server-2019.html?lcid=it&culture=it-it&country=it>

Per avere una gestione completa di Sql Server, si può inoltre installare Sql Server Management Studio al seguente link:

<https://go.microsoft.com/fwlink/?linkid=2203303&clcid=0x409>

Successivamente, per accedere al server bisognerà aprire installare Sql Server Management Studio ed effettuare i passi in figura:

Immagine che contiene testo

Descrizione generata automaticamente

La struttura del progetto, contenente file e cartelle, è la seguente:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Per poter salvare lo stato della sessione su Sql Server, è necessario cambiare il file Web.config, aggiungendo informazioni e parametri tra i tag *SessionState:*

*<sessionState*

*mode="SQLServer"*

*sqlConnectionString="Data Source=.;*

*Initial Catalog=TestSession;*

*Persist Security Info=True;*

*User ID=user;Password=######## "*

*timeout="100"*

*cookieless="false"*

*allowCustomSqlDatabase="true"*

*customProvider="DefaultSessionProvider">*

*<providers>*

*<add name="DefaultSessionProvider" type="System.Web.Providers.DefaultSessionStateProvider,*

*System.Web.Providers, Version=1.0.0.0, Culture=neutral,*

*PublicKeyToken=31bf3856ad364e35"*

*connectionStringName="tempdbConnectionString1" />*

*</providers>*

</sessionState>

La modalità SQLServer garantisce che lo stato della sessione venga mantenuto se l'applicazione Web viene riavviata e rende anche disponibile lo stato della sessione a più server Web della Web farm creata nel capitolo 3.

Per cambiare i permessi per accedere alla pagina Web del server è necessario cambiare l’impostazione i tipo d’identità nella sezione Modello di processo:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Successivamente, per creare lo stato della sessione su Sql Server sul prompt dei comandi, nella path

*C:\Windows\Microsoft.NET\Framework64\version*

E digitare il comando seguente:  
*aspnet\_regsql.exe -S . -U user -P passwordDatabase -d TestSession -ssadd -sstype c*

Come mostrato in figura:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Successivamente, su SQL Server verrà creato il database TestSession, per cambiare il nome basterà andare sul dbo TempGetAddId e cambiare l’appname sul nome desiderato:

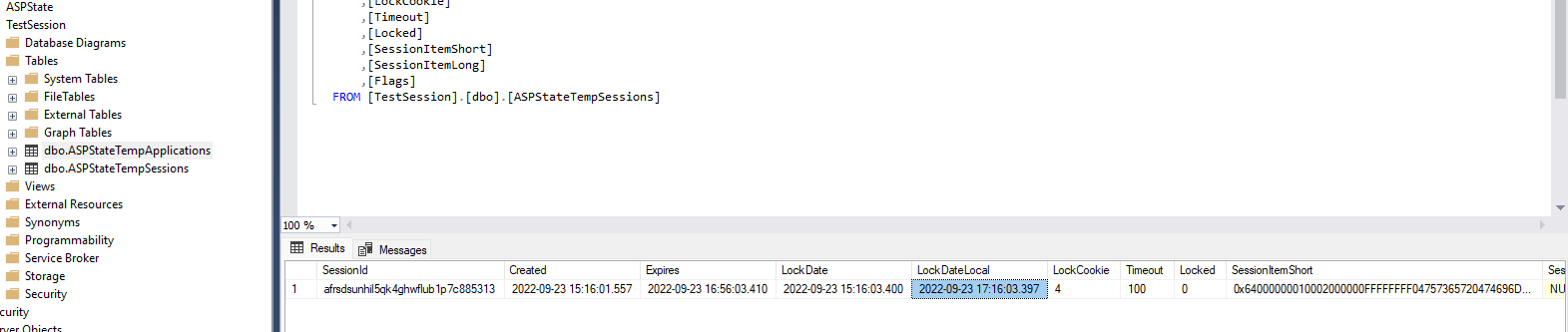
Figura

Immagine che contiene testo

Descrizione generata automaticamente

Lo stato della sessione dell’applicazione verrà così salvata temporaneamente, poiché il time-out della sessione è di cento ottanta minuti, cioè quelli impostati nel file Web.config dell’applicazione in Asp.NET .

Figura



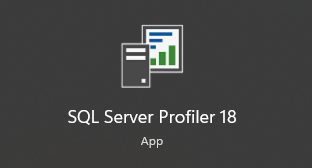
# Analisi delle prestazioni

## **Test delle prestazioni con SQL SERVER**

Sql Server mette a disposizione un tool per analizzare la performance dei database, ma ci sono anche vari tool esterni.

Per il caso di studio in questione verranno eseguiti test di tipo di performance con “Microsoft Sql Server Profiler”.

Figura



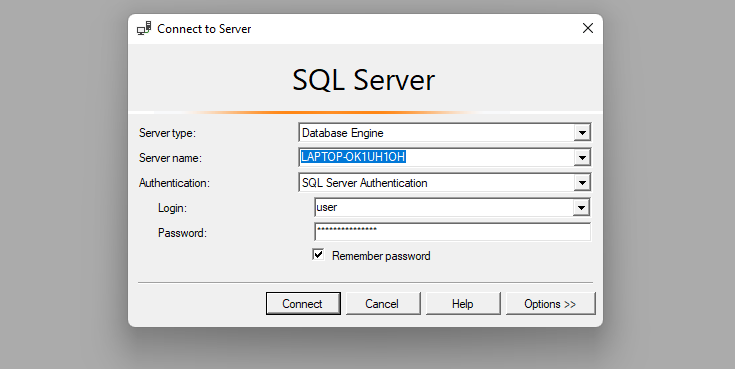
SQL Server Profiler è incluso nel software SQL Server e fornisce informazioni riguardo query SQL, procedure di archiviazione, procedure remota, eccetera.

Inoltre, SQL può fornire dei trackfiles basati sulla configurazione dell'utente, come gli utenti hanno scelto di configurare le proprietà del profiler, impostando le zone su cui sono maggiormente interessati a testare le performance.

Fornisce inoltre flessibilità agli utenti: è infatti possibile scegliere su quale dei vari database del server si andranno ad effettuare i Performance test.

Per prima cosa, dopo aver aperto Sql Profiler è necessario fornire le credenziali di accesso alla connessione del database:

Figura



Poi s’imposta il nome del file traccia, e s’imposta la path in cui salvare il file:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Succesivamente, nella sesione Events Selections, cliccando “Show all events e “Show all columns”, si possono selezionare tutti gli eventi di cui tenere traccia.

In questo caso interessano le performance, quindi si selezionano tutti i campi relativi ad essa:

Figura

Immagine che contiene testo, tavolo

Descrizione generata automaticamente

Dato che, non è interessante conoscere tutti i singoli eventi, ma solo quelli con un certo tempo di risposta rilevante (500 millisecondi ad esempio), si clicca su “Column filters” e si selezionano i filtri:

* DatabaseId = XX –database che si vuole testare dalla query select \* from sys.databases query
* Duration >= 500 (in milliseconds)
* CPU >= 500 (in milliseconds)

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Infine, si preme ok e poi Run.

D’ora in poi, qualsiasi query fatta sul database che abbia un tempo di risposta maggiore di 500 millisecondi verrà catturato nello stacktrace.

La stack trace generata è questa:

Figura

Immagine che contiene tavolo

Descrizione generata automaticamente

Si può notare, nella colonna Cpu e Duration, il tempo, espresso in millisecondi, passato a tracciare l’evento. In questo caso quindi per l’evento “Performance Statistic” il tempo impiegato va dai 3 ai 14 millisecondi, quindi relativamente veloce.

Gli eventi registrati nello stack trace possono essere decine, (se non centinaia o migliaia), analizzarli sarebbe troppo complicato.

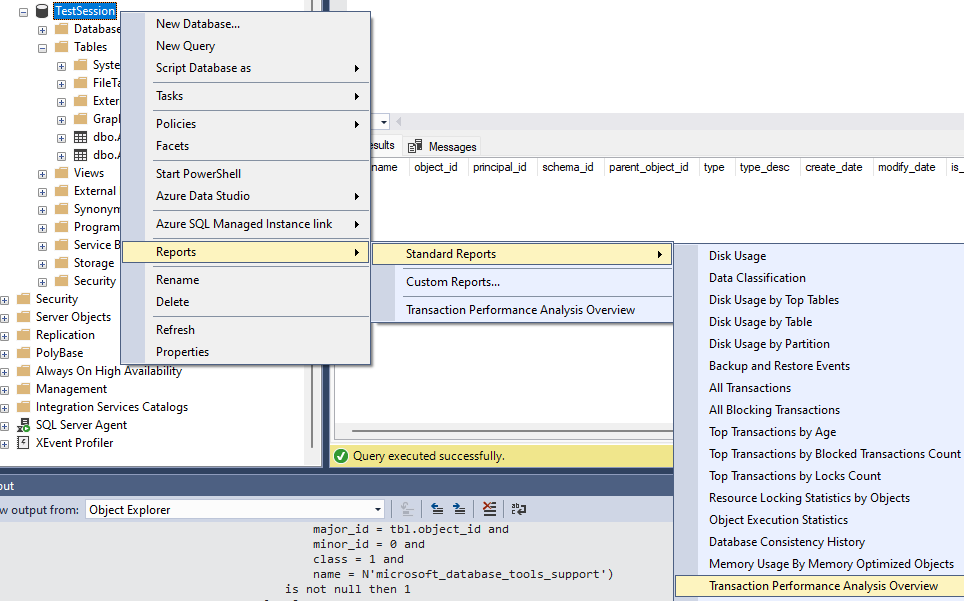
Ma c’è la possibilità di inserire tutti questi dati nella tabella del database e, una volta fatto, lavorando sulla SQL si possono trovare informazioni utili.

Dopo aver eseguito la query SQL su di esso, recuperando le informazioni utili si possono ottimizzare le tabelle per ottimizzare le prestazioni del database.

È possibile, oltre a questi test, andare ad effettuare test sul database direttamente da Microsoft SQL Server Management Studio.

Per farlo, basta cliccare con il tasto destro, sul database a scelta, e poi selezionare Reports 🡪 Standard Reports 🡪 Transaction Performance Analysis Overview, come in figura:

Figura



Dopodiché, si seleziona “Tables Analysis”:

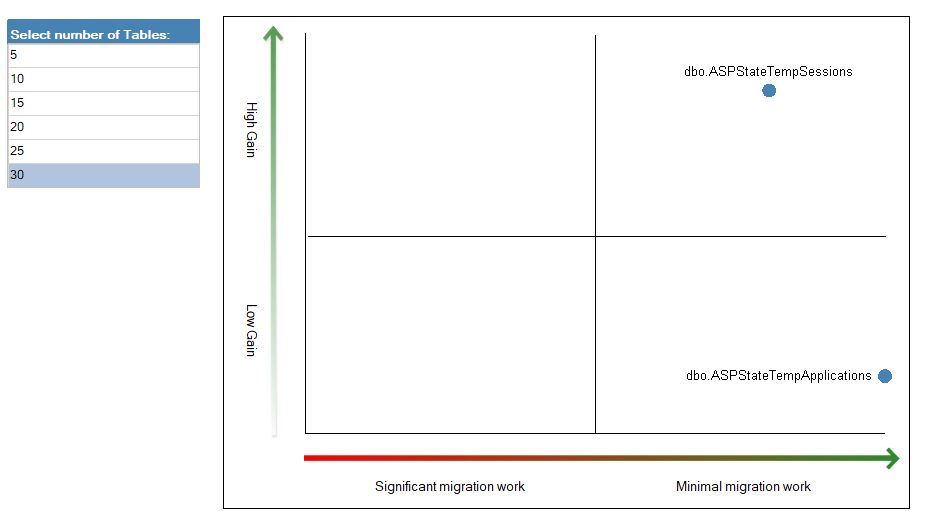
Figura

Immagine che contiene testo

Descrizione generata automaticamente

Verrà generato un grafico rappresentante le performance:

Grafico



Questo grafico indica le principali tabelle candidate per l'ottimizzazione della memoria in base ai modelli di accesso del carico di lavoro.

**L'asse orizzontale** rappresenta lo **sforzo decrescente di ottimizzazione della memoria**, mentre **l'asse verticale** rappresenta **i vantaggi crescenti dell'ottimizzazione della memoria nel carico di lavoro**.

Le tabelle in alto a destra sono quelle a cui bisognerebbe dare la priorità per ottimizzare l’uso della memoria.

## **Test delle prestazioni con REDIS**

Redis include il tool redis-benchmark, che può simulare vari comandi per testare un certo numero di client che eseguano un certo numero di query.

Per analizzare le performance di Redis, basta andare sulla path d’installazione (C:\Program Files\Redis) e cliccare su redis-benchmark:

Figura

Immagine che contiene testo

Descrizione generata automaticamente

Redis inizierà a fare i test sulle performance stampandoli sul prompt dei comandi.

Il risultato dell’esecuzione è la seguente lista di risultati:

PING\_INLINE: -1.#PING\_INLINE: 51520.00PING\_INLINE: 51128.00PING\_INLINE: 51261.33PING\_INLINE: 51388.00PING\_INLINE: 51455.20PING\_INLINE: 51552.00PING\_INLINE: 51590.29====== PING\_INLINE ======**100000 requests completed in 1.94 seconds #REDIS ha completato 100000 richieste in soli 1.94 secondi.**

50 parallel clients 3 bytes payload keep alive: 189.74% <= 1 milliseconds100.00% <= 1 milliseconds51493.30 requests per secondPING\_BULK: 50441.86PING\_BULK: 49849.83PING\_BULK: 50407.00PING\_BULK: 49791.93PING\_BULK: 50023.01PING\_BULK: 50558.39PING\_BULK: 50823.72PING\_BULK: 50373.68====== PING\_BULK ====== **100000 requests completed in 1.98 seconds** 50 parallel clients 3 bytes payload keep alive: 188.23% <= 1 milliseconds99.83% <= 2 milliseconds100.00% <= 2 milliseconds50505.05 requests per secondSET: 49229.17SET: 51342.28SET: 51226.28SET: 51100.25SET: 51223.29SET: 50948.38SET: 50967.70SET: 50622.91====== SET ====== **100000 requests completed in 1.99 seconds** 50 parallel clients 3 bytes payload keep alive: 184.49% <= 1 milliseconds99.90% <= 2 milliseconds100.00% <= 2 milliseconds50352.47 requests per secondGET: 50914.89GET: 51979.80GET: 52489.95GET: 52912.17GET: 52904.49GET: 52945.26GET: 52906.91GET: 54091.26====== GET ====== **100000 requests completed in 1.85 seconds** 50 parallel clients 3 bytes payload keep alive: 194.35% <= 1 milliseconds100.00% <= 1 milliseconds54054.05 requests per secondINCR: 52186.81INCR: 52321.76INCR: 51837.25INCR: 51081.55INCR: 51104.06INCR: 51108.24INCR: 51262.19====== INCR ====== **100000 requests completed in 1.90 seconds** 50 parallel clients 3 bytes payload keep alive: 189.04% <= 1 milliseconds100.00% <= 1 milliseconds52603.89 requests per secondLPUSH: 49687.50LPUSH: 50849.63LPUSH: 50292.64LPUSH: 50771.54LPUSH: 50399.61LPUSH: 48645.34LPUSH: 47858.84LPUSH: 47237.26LPUSH: 46699.40====== LPUSH ====== **100000 requests completed in 2.18 seconds** 50 parallel clients 3 bytes payload keep alive: 157.63% <= 1 milliseconds99.73% <= 2 milliseconds100.00% <= 2 milliseconds45766.59 requests per secondLPOP: 27649.12LPOP: 26263.84LPOP: 36324.96LPOP: 41045.85LPOP: 43004.73LPOP: 44409.33LPOP: 45304.43LPOP: 46052.57LPOP: 46215.85====== LPOP ====== **100000 requests completed in 2.16 seconds** 50 parallel clients 3 bytes payload keep alive: 172.29% <= 1 milliseconds97.72% <= 2 milliseconds99.89% <= 3 milliseconds100.00% <= 3 milliseconds46339.20 requests per secondSADD: 48477.61SADD: 49322.92SADD: 48649.84SADD: 48981.90SADD: 49474.43SADD: 49447.26SADD: 49414.93SADD: 49621.02====== SADD ====== **100000 requests completed in 2.01 seconds** 50 parallel clients 3 bytes payload keep alive: 180.50% <= 1 milliseconds99.85% <= 2 milliseconds100.00% <= 2 milliseconds49751.24 requests per secondSPOP: 52348.63SPOP: 52774.38SPOP: 52802.95SPOP: 52833.53SPOP: 52362.49SPOP: 52474.61SPOP: 52513.36SPOP: 52356.11====== SPOP ====== **100000 requests completed in 1.91 seconds** 50 parallel clients 3 bytes payload keep alive: 193.07% <= 1 milliseconds100.00% <= 1 milliseconds52328.63 requests per secondLPUSH (needed to benchmark LRANGE): 49907.61LPUSH (needed to benchmark LRANGE): 50096.78LPUSH (needed to benchmark LRANGE): 50485.38LPUSH (needed to benchmark LRANGE): 50433.62LPUSH (needed to benchmark LRANGE): 50448.48LPUSH (needed to benchmark LRANGE): 50492.33LPUSH (needed to benchmark LRANGE): 50454.27LPUSH (needed to benchmark LRANGE): 50490.18====== LPUSH (needed to benchmark LRANGE) ====== **100000 requests completed in 1.98 seconds** 50 parallel clients 3 bytes payload keep alive: 185.91% <= 1 milliseconds100.00% <= 2 milliseconds50505.05 requests per secondLRANGE\_100 (first 100 elements): 25989.47LRANGE\_100 (first 100 elements): 26118.18LRANGE\_100 (first 100 elements): 26179.71LRANGE\_100 (first 100 elements): 26148.94LRANGE\_100 (first 100 elements): 26136.13LRANGE\_100 (first 100 elements): 26082.64LRANGE\_100 (first 100 elements): 26079.29LRANGE\_100 (first 100 elements): 26026.29LRANGE\_100 (first 100 elements): 26020.09LRANGE\_100 (first 100 elements): 26018.44LRANGE\_100 (first 100 elements): 26038.66LRANGE\_100 (first 100 elements): 26043.54LRANGE\_100 (first 100 elements): 26055.49LRANGE\_100 (first 100 elements): 26071.22LRANGE\_100 (first 100 elements): 26057.99====== LRANGE\_100 (first 100 elements) ====== **100000 requests completed in 3.84 seconds** 50 parallel clients 3 bytes payload keep alive: 158.51% <= 1 milliseconds99.92% <= 2 milliseconds100.00% <= 3 milliseconds26028.11 requests per secondLRANGE\_300 (first 300 elements): 12120.48LRANGE\_300 (first 300 elements): 11529.94LRANGE\_300 (first 300 elements): 11529.11LRANGE\_300 (first 300 elements): 11656.29LRANGE\_300 (first 300 elements): 11569.58LRANGE\_300 (first 300 elements): 11479.04LRANGE\_300 (first 300 elements): 11616.26LRANGE\_300 (first 300 elements): 11675.56LRANGE\_300 (first 300 elements): 11572.59LRANGE\_300 (first 300 elements): 11645.85LRANGE\_300 (first 300 elements): 11699.38LRANGE\_300 (first 300 elements): 11726.92LRANGE\_300 (first 300 elements): 11667.21LRANGE\_300 (first 300 elements): 11731.14LRANGE\_300 (first 300 elements): 11769.64LRANGE\_300 (first 300 elements): 11814.06LRANGE\_300 (first 300 elements): 11774.38LRANGE\_300 (first 300 elements): 11773.15LRANGE\_300 (first 300 elements): 11730.68LRANGE\_300 (first 300 elements): 11767.96LRANGE\_300 (first 300 elements): 11683.35LRANGE\_300 (first 300 elements): 11684.19LRANGE\_300 (first 300 elements): 11716.35LRANGE\_300 (first 300 elements): 11744.44LRANGE\_300 (first 300 elements): 11747.58LRANGE\_300 (first 300 elements): 11746.65LRANGE\_300 (first 300 elements): 11717.49LRANGE\_300 (first 300 elements): 11684.77LRANGE\_300 (first 300 elements): 11693.94LRANGE\_300 (first 300 elements): 11711.20LRANGE\_300 (first 300 elements): 11734.28LRANGE\_300 (first 300 elements): 11739.56LRANGE\_300 (first 300 elements): 11744.88LRANGE\_300 (first 300 elements): 11761.13====== LRANGE\_300 (first 300 elements) ====== **100000 requests completed in 8.50 seconds** 50 parallel clients 3 bytes payload keep alive: 10.00% <= 1 milliseconds36.96% <= 2 milliseconds99.50% <= 3 milliseconds99.98% <= 4 milliseconds100.00% <= 4 milliseconds11763.32 requests per secondLRANGE\_500 (first 450 elements): 8670.59LRANGE\_500 (first 450 elements): 8871.64LRANGE\_500 (first 450 elements): 8962.46LRANGE\_500 (first 450 elements): 8966.51LRANGE\_500 (first 450 elements): 8956.76LRANGE\_500 (first 450 elements): 8905.90LRANGE\_500 (first 450 elements): 8774.21LRANGE\_500 (first 450 elements): 8766.97LRANGE\_500 (first 450 elements): 8782.61LRANGE\_500 (first 450 elements): 8790.71LRANGE\_500 (first 450 elements): 8756.26LRANGE\_500 (first 450 elements): 8776.69LRANGE\_500 (first 450 elements): 8791.87LRANGE\_500 (first 450 elements): 8796.06LRANGE\_500 (first 450 elements): 8810.56LRANGE\_500 (first 450 elements): 8822.12LRANGE\_500 (first 450 elements): 8833.98LRANGE\_500 (first 450 elements): 8847.89LRANGE\_500 (first 450 elements): 8857.27LRANGE\_500 (first 450 elements): 8867.09LRANGE\_500 (first 450 elements): 8878.33LRANGE\_500 (first 450 elements): 8887.96LRANGE\_500 (first 450 elements): 8897.79LRANGE\_500 (first 450 elements): 8904.88LRANGE\_500 (first 450 elements): 8910.42LRANGE\_500 (first 450 elements): 8916.14LRANGE\_500 (first 450 elements): 8920.83LRANGE\_500 (first 450 elements): 8925.17LRANGE\_500 (first 450 elements): 8931.34LRANGE\_500 (first 450 elements): 8934.89LRANGE\_500 (first 450 elements): 8939.15LRANGE\_500 (first 450 elements): 8941.73LRANGE\_500 (first 450 elements): 8944.27LRANGE\_500 (first 450 elements): 8947.85LRANGE\_500 (first 450 elements): 8950.99LRANGE\_500 (first 450 elements): 8948.65LRANGE\_500 (first 450 elements): 8948.97LRANGE\_500 (first 450 elements): 8947.04LRANGE\_500 (first 450 elements): 8950.09LRANGE\_500 (first 450 elements): 8952.67LRANGE\_500 (first 450 elements): 8955.04LRANGE\_500 (first 450 elements): 8956.90LRANGE\_500 (first 450 elements): 8959.14LRANGE\_500 (first 450 elements): 8961.65LRANGE\_500 (first 450 elements): 8963.95====== LRANGE\_500 (first 450 elements) ====== **100000 requests completed in 11.16 seconds** 50 parallel clients 3 bytes payload keep alive: 10.00% <= 1 milliseconds0.09% <= 2 milliseconds89.40% <= 3 milliseconds99.79% <= 4 milliseconds99.98% <= 5 milliseconds100.00% <= 5 milliseconds8963.79 requests per secondLRANGE\_600 (first 600 elements): 7376.24LRANGE\_600 (first 600 elements): 7334.80LRANGE\_600 (first 600 elements): 7282.67LRANGE\_600 (first 600 elements): 7256.01LRANGE\_600 (first 600 elements): 7237.39LRANGE\_600 (first 600 elements): 7193.28LRANGE\_600 (first 600 elements): 7181.29LRANGE\_600 (first 600 elements): 7185.02LRANGE\_600 (first 600 elements): 7174.50LRANGE\_600 (first 600 elements): 7173.30LRANGE\_600 (first 600 elements): 7173.54LRANGE\_600 (first 600 elements): 7181.85LRANGE\_600 (first 600 elements): 7190.30LRANGE\_600 (first 600 elements): 7198.39LRANGE\_600 (first 600 elements): 7204.84LRANGE\_600 (first 600 elements): 7205.19LRANGE\_600 (first 600 elements): 7209.38LRANGE\_600 (first 600 elements): 7212.43LRANGE\_600 (first 600 elements): 7212.74LRANGE\_600 (first 600 elements): 7213.27LRANGE\_600 (first 600 elements): 7215.69LRANGE\_600 (first 600 elements): 7218.27LRANGE\_600 (first 600 elements): 7219.18LRANGE\_600 (first 600 elements): 7220.88LRANGE\_600 (first 600 elements): 7215.87LRANGE\_600 (first 600 elements): 7211.18LRANGE\_600 (first 600 elements): 7209.09LRANGE\_600 (first 600 elements): 7203.32LRANGE\_600 (first 600 elements): 7200.80LRANGE\_600 (first 600 elements): 7198.08LRANGE\_600 (first 600 elements): 7193.04LRANGE\_600 (first 600 elements): 7191.38LRANGE\_600 (first 600 elements): 7188.24LRANGE\_600 (first 600 elements): 7189.25LRANGE\_600 (first 600 elements): 7187.25LRANGE\_600 (first 600 elements): 7182.69LRANGE\_600 (first 600 elements): 7180.84LRANGE\_600 (first 600 elements): 7182.45LRANGE\_600 (first 600 elements): 7183.41LRANGE\_600 (first 600 elements): 7177.92LRANGE\_600 (first 600 elements): 7176.48LRANGE\_600 (first 600 elements): 7174.05LRANGE\_600 (first 600 elements): 7172.87LRANGE\_600 (first 600 elements): 7172.47LRANGE\_600 (first 600 elements): 7168.41LRANGE\_600 (first 600 elements): 7167.33LRANGE\_600 (first 600 elements): 7168.68LRANGE\_600 (first 600 elements): 7169.89LRANGE\_600 (first 600 elements): 7168.76LRANGE\_600 (first 600 elements): 7165.43LRANGE\_600 (first 600 elements): 7165.75LRANGE\_600 (first 600 elements): 7164.61LRANGE\_600 (first 600 elements): 7164.82LRANGE\_600 (first 600 elements): 7163.44LRANGE\_600 (first 600 elements): 7164.31====== LRANGE\_600 (first 600 elements) ====== **100000 requests completed in 13.96 seconds** 50 parallel clients 3 bytes payload keep alive: 10.00% <= 1 milliseconds0.02% <= 2 milliseconds7.56% <= 3 milliseconds95.01% <= 4 milliseconds99.96% <= 5 milliseconds100.00% <= 6 milliseconds100.00% <= 6 milliseconds7164.86 requests per secondMSET (10 keys): 42620.00MSET (10 keys): 44200.00MSET (10 keys): 44709.09MSET (10 keys): 44737.50MSET (10 keys): 44726.67MSET (10 keys): 44860.00MSET (10 keys): 44762.58MSET (10 keys): 44648.34MSET (10 keys): 44252.20====== MSET (10 keys) ====== **100000 requests completed in 2.27 seconds** 50 parallel clients 3 bytes payload keep alive: 136.79% <= 1 milliseconds99.89% <= 2 milliseconds99.90% <= 4 milliseconds99.95% <= 7 milliseconds99.95% <= 8 milliseconds100.00% <= 8 milliseconds44052.86 requests per second

Analizzando I risultati e costruendo un grafico basato sul tempo di esecuzione in base al numero di richieste, si ottiene:

Grafico 2

Inoltre, è possibile specificare l’host, la porta, il tipo e il numero di richieste al server Redis, attraverso il comando:

*redis-benchmark -h 127.0.0.1 -p 6379 -t set,lpush -n 1000000 -q*

Figura 48

Immagine che contiene testo

Descrizione generata automaticamente

Ovviamente, il numero di richieste che si possono gestire al secondo dipende molto dall’architettura hardware della macchina su cui è utilizzato Redis.

Nella presentazione <https://www.slideshare.net/devopsguys/best-performing-aspnet-session-state-providers>, l’autore TheDevMgr ha testato le migliori performance in diversi provider in cui si è andato a memorizzare lo stato della sessione di asp.Net.

I Server sono stati:

* Asp.net in proc
* Asp.net state server
* Redis
* Couchbase
* RavenDb
* Microsoft SQL Server 2012
* Mongo DB.

L’applicazione era formata da due Web view: una semplice, ed una complessa.

Quella semplice formata da tipi di dato, come stringhe, integer, double e datetimes,

quella complessa da liste generiche, array lists, dataset e degli oggetti personalizzati.

Nella Web View semplice sono stati usati trenta oggetti in sessione, in quella complessa quaranta oggetti.

**I risultati:**

Figura

Immagine che contiene tavolo

Descrizione generata automaticamente

I tempi di risposta dei server sono stati:

Figura

Immagine che contiene tavolo

Descrizione generata automaticamente

Come si può evincere , Redis risulta il migliore in quasi tutti i casi, tranne nella gestione della memoria, in cui il migliore è stato SQL Server 2012, e nel tempo di risposta con Asp.net state con modalità in-proc (questa modalità archivia i dati della sessione in un oggetto di memoria nel processo dell'applicazione: aspnet\_wp.exe).

# Conclusioni e Sviluppi Futuri

In conclusione, si è visto come memorizzare lo stato della sessione di un’applicazione sia con l’utilizzo di Redis, sia con l’utilizzo di SQL Server.

Si è notato, innanzitutto, come sia più semplice salvare lo stato della sessione dell’applicazione su Redis rispetto a SQL Server, ma soprattutto, come il carico di lavoro di Redis (quindi il numero di richieste) sia gestito meglio.

SQL server è comunque un database relazionale, mentre Redis non lo è.

Questo fattore potrebbe essere, insieme ad altri, il motivo per cui Redis è più prestante rispetto a Sql Server (ma anche rispetto a molti altri, come già visto).

Come sviluppi futuri, si potrebbe aumentare il cluster di dati da memorizzare nella sessione, andando a studiare come cambiano le performance all’aumentare del numero di oggetti, che andamento ha (esponenziale, lineare o di altro tipo).

Il codice utilizzato per poter replicare i risultati è disponibile al seguente link:

<https://github.com/matteotoma98/Tesi>

# Bibliografia

[Sou2022], “Redis”, 2022,

<https://en.wikipedia.org/wiki/Redis>

[Tde2022], “Redis Streams”, 2022,

<https://redis.io/docs/data-types/streams/>

[RicKtr2022], Rick A., Ktrnthsnr et. al, “Panoramica di ASP.NET”, 2022,

<https://docs.microsoft.com/it-it/aspnet/overview>

[Fiy2015] Fiyaz H., “ASP.NET Session State Management with Redis (Local Server Farm Testing)”, 2015, <https://www.codeproject.com/Articles/1040453/ASP-NET-Session-State-Management-with-Redis-Local>

[Vai2020] Vaidy G., “IIS Express Overview”, 2020,

<https://docs.microsoft.com/it-it/iis/extensions/introduction-to-iis-express/iis-express-overview>

[Lin2019] Linda R., “DEFINITION Internet Information Services (IIS)”, 2019, <https://www.techtarget.com/searchwindowsserver/definition/IIS>

[Rus42020] Ruslan Y., “Using the URL Rewrite Module”, 2020, <https://docs.microsoft.com/en-us/iis/extensions/url-rewrite-module/using-the-url-rewrite-module>

[HaiSev2022] “Configure SQL Server to store ASP.NET session state”, 2022,

<https://learn.microsoft.com/en-us/troubleshoot/developer/webapps/aspnet/development/configure-sql-store-session-state>

[Rup], RUPESH S., “Testing the performance of a database”, <https://www.qentelli.com/thought-leadership/insights/ms-sql-server-profiler-how-does-it-help-testing-performance-database>

[BanMil], Kyle B., Nermina M., “Redis benchmark,”2022,

<https://redis.io/docs/reference/optimization/benchmarks/>

[DevOps], DevOps Guys, “Best performing asp.net session state providers”, 2013, <https://www.slideshare.net/devopsguys/best-performing-aspnet-session-state-providers>

“Che cos’è ASP.NET?” , 2022,

<https://www.ionos.it/digitalguide/siti-web/programmazione-del-sito-web/cose-aspnet/>

<https://redis.io/>

“Introduction to ASP.NET”, 2019,

<https://www.geeksforgeeks.org/introduction-to-asp-net/>

“Cos'è Redis?”,

<https://aws.amazon.com/it/redis/>

“What Is Round-Robin Load Balancing?”,

<https://www.nginx.com/resources/glossary/round-robin-load-balancing/>

Session-State Modes, 2014,

<https://learn.microsoft.com/en-us/previous-versions/ms178586(v=vs.140)?redirectedfrom=MSDN>

# Elenco delle figure

[Figura 1 7](#_Toc115798885)

[Figura 2 11](#_Toc115798886)

[Figura 3 13](#_Toc115798887)

[Figura 4 15](#_Toc115798888)

[Figura 5 15](#_Toc115798889)

[Figura 6 16](#_Toc115798890)

[Figura 7 17](#_Toc115798891)

[Figura 8 18](#_Toc115798892)

[Figura 9 18](#_Toc115798893)

[Figura 10 19](#_Toc115798894)

[Figura 11 20](#_Toc115798895)

[Figura 12 20](#_Toc115798896)

[Figura 13 21](#_Toc115798897)

[Figura 14 21](#_Toc115798898)

[Figura 15 21](#_Toc115798899)

[Figura 16 22](#_Toc115798900)

[Figura 17 23](#_Toc115798901)

[Figura 18 24](#_Toc115798902)

[Figura 19 24](#_Toc115798903)

[Figura 20 25](#_Toc115798904)

[Figura 21 25](#_Toc115798905)

[Figura 22 26](#_Toc115798906)

[Figura 23 30](#_Toc115798907)

[Figura 24 31](#_Toc115798908)

[Figura 25 31](#_Toc115798909)

[Figura 26 32](#_Toc115798910)

[Figura 27 32](#_Toc115798911)

[Figura 28 33](#_Toc115798912)

[Figura 29 33](#_Toc115798913)

[Figura 30 35](#_Toc115798914)

[Figura 31 36](#_Toc115798915)

[Figura 32 36](#_Toc115798916)

[Figura 33 37](#_Toc115798917)

[Figura 34 39](#_Toc115798918)

[Figura 35 41](#_Toc115798919)

[Figura 36 41](#_Toc115798920)

[Figura 37 42](#_Toc115798921)

[Figura 38 42](#_Toc115798922)

[Figura 39 43](#_Toc115798923)

[Figura 40 44](#_Toc115798924)

[Figura 41 44](#_Toc115798925)

[Figura 42 45](#_Toc115798926)

[Figura 43 46](#_Toc115798927)

[Figura 44 46](#_Toc115798928)

[Figura 45 47](#_Toc115798929)

[Figura 46 48](#_Toc115798930)

[Figura 47 50](#_Toc115798931)

[Figura 48 64](#_Toc115798932)

[Figura 49 65](#_Toc115798933)

[Figura 50 66](#_Toc115798934)

# Elenco dei grafici

[Grafico 1 48](#_Toc115798947)

[Grafico 2 64](#_Toc115798948)

# Ringraziamenti

Un ringraziamento speciale va a tutta la mia famiglia, che mi ha sostenuto incessantemente tutti gli anni, senza mai dubitare di me e incoraggiandomi sempre.

Un altro ringraziamento va ad Amal e Federico, che mi hanno sempre aiutato a distrarmi dai problemi e dalle giornate pesanti, oltre a partecipare ai vari progetti durante la triennale.

Ringrazio i miei coinquilini, soprattutto Giorgio, che per un anno mi ha sopportato e supportato in ogni mio sfogo.

Ringrazio il mio relatore Fausto Marcantoni e il mio correlatore Roberto Gagliardi per aver creduto in me e condiviso questo argomento.

Infine, ringrazio tutti i Professori che mi hanno insegnato moltissime cose, più di quanto potessi immaginare, e sono stati molto bravi nel farlo, motivandomi a scegliere questa complicata e affascinante materia.

Dedico questo traguardo ai miei genitori e ai miei parenti, che mi hanno cresciuto e visto maturare in questi anni fino a questo momento della mia vita.

Matteo Toma, 26/10/22