Strategie di condivisione dati tra istanze nel Cloud

#### di [Roberto Freato](https://mvp.support.microsoft.com/profile=9F9B3C0A-2016-4034-ACD6-9CEDEE74FAF3) – Microsoft MVP

* 1. 

*Luglio, 2012*

In questo articolo verranno discussi i seguenti argomenti:

* + Tecniche di condivisione dati in ambienti distribuiti
  + Esternalizzazione dello storage dalle istanze Cloud
  + Internalizzazione dello storage nelle istanze Cloud

E le seguenti tecnologie:

* + Windows Azure Cloud Services (Nota 1)
  + Windows Azure Storage
  + SMB File Sharing
  + Command Prompt ☺

## Sommario

* + [Scenario 1 – la nuova applicazione](#_Scenario_1_–)
  + [Scenario 2 – la migrazione](#_Scenario_2_–)
    - [Soluzione di base](#_Soluzione_di_base)
    - [Soluzione di alta disponibilità e affidabilità](#_Soluzione_di_alta)
  + [Approfondimenti](#_Approfondimenti)

In questo articolo parleremo delle strategie più diffuse di condividere i dati tra i vari attori in gioco nello scenario Cloud-enabled, ove per attori in gioco intendiamo i servizi (e più a basso livello, le macchine virtuali) che compongono la nostra architettura distribuita.

Nel Cloud di Azure esistono sostanzialmente (per il momento) tre scenari di esecuzione:

* 1. **Nota:** Il 7 Giugno 2012 sono state presentate moltissime novità che rivoluzionano l’offerta della piattaforma in termini di tipologia di servizi (IaaS, Azure Web Sites, Caching Roles) che non trattiamo in questo articolo in quanto trattasi di servizi in CTP e/o comunque di interesse principale da parte dei super-early adopters.
  2. Esecuzione nel contesto di una WebRole, ovvero di una istanza di macchina virtuale specificatamente concepita per ospitare una serie di applicazioni/servizi web-based (a cura di IIS).
  3. Esecuzione nel contesto di un WorkerRole, ovvero di una istanza molto “leggera” specificatamente concepita per ospitare processi di computazione/esecuzione long-running, servizi di background e processi di gestione del ciclo di vita.
  4. Esecuzione nel contesto di una VMRole, particolare tipo di istanza sempre meno supportata e sempre più “evitata” (in quanto ormai retaggio della prossima offerta di IaaS), dedicata a coloro volessero personalizzare il comportamento del proprio PaaS.

In questi scenari, gli unici in cui (al momento) viene eseguito del codice applicativo del cliente/utente, è spesso necessario avere un repository condiviso per memorizzare dati. Per questo parleremo di alcuni scenari tipici con le loro tipiche soluzioni “consigliate”, dove il consiglio arriva più per una applicazione di esperienza sul campo che per la effettiva presenza di una best-practice nella letteratura di settore.

* 1. **Nota 1:** Una doverosa premessa visto che da poco è noto pubblicamente che è in corso un renaming di alcuni servizi legati alla Windows Azure Platform. In particolare una email dell’8 Maggio a tutti i clienti di Azure specifica il cambio di nome per i vari servizi di Azure al solo scopo di semplificare la lettura nell’area billing. Tuttavia, concordemente ai vari rumors che hanno seguito, non escludo che dalla scrittura di questo articolo alla sua pubblicazione e nell’immediato futuro ad essa, i nomi utilizzati in questo articolo potrebbero aver perso in parte validità.

## Scenario 1 – la nuova applicazione

Nelle nuove applicazioni il concetto di adattamento non esiste o meglio, non dovrebbe. Una soluzione veramente “nuova” dovrebbe poter stabilire “from scratch” i punti salienti della propria architettura senza forti limitazioni. Il posto in cui poi si piazzerà il risultato e l’ambiente che gli si creerà intorno deve essere, a mio avviso, approssimabile ad un dettaglio implementativo.

Per questo, quando si vuole approcciare una nuova soluzione siamo più liberi (storicamente) di avanzare ipotesi più ardite (dal punto di vista dell’innovazione) corroborate da una progettazione sinergica dei vari strumenti messi a disposizione dalla tecnologia. In uno scenario Cloud-based questi strumenti tuttavia sono spesso anche “vendor-based” o “environment-based” e quindi, generando potenziale lock-in, rompono la supposizione appena fatta di poter lavorare in condizioni di astrazione dall’ambiente di esecuzione.

Per cui sovviene la domanda: “che vincoli ci sono per le applicazioni in Azure?” oppure “che vincoli ci sono per le applicazioni nel cloud?” o ancora “che vincoli ci sono per le applicazioni in una generica web-farm distribuita?”. La risposta è mediamente complessa, per cui decliniamola solamente allo scopo di questo articolo, ovvero all’accesso ai dati.

La necessità nel cloud è di esternalizzare il dato dalla singola istanza per il motivo banale che dal momento in cui potrebbero esserci più istanze, ognuna andrebbe a lavorare sul dato locale perdendo lo stato globale dell’applicazione. Potrei cambiare il discorso dicendo che il 90% dei problemi del cloud sono legati ad esternalizzare i dati dalla logica delle soluzioni, per rendere meglio l’idea.

Partendo dal nuovo, possiamo perciò affidarci alla tecnologia di storage del vendor (con le opportune precauzioni), che nel caso di Azure è l’Azure Storage, una insieme di tre servizi basati su REST, finalizzati alla memorizzazione massiva di dati, di cui un esempio di codice è presente sotto:

* 1. public Uri UploadBlob(string path, string fileName, string content)
  2. {
  3. var account = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");
  4. var cloudBlobClient = account.CreateCloudBlobClient();
  5. var cloudBlobContainer = cloudBlobClient.GetContainerReference(path);
  6. cloudBlobContainer.CreateIfNotExist();
  7. var blob = cloudBlobContainer.GetBlobReference(fileName);
  8. blob.UploadText(content);
  9. return blob.Uri;
  10. }

Trattandosi di un meccanismo custom di salvataggio di files, nelle nuove soluzioni è consigliato comunque limitare al minimo il numero di punti nel codice in cui se ne fa utilizzo, riusando il più possibile lo stesso codice per permettere, in caso di “move-out” il facile adattamento ad un diverso sistema di storage, riducendo al minimo il lock-in.

* 1. **Nota:** Una buona architettura dovrebbe utilizzare in modo intelligente le potenzialità di una implementazione, minimizzando al contempo il rischio di lock-in proporzionalmente alla possibilità reale di dover o voler cambiare tecnologia in un secondo momento.

Non sempre tuttavia si lavora su nuove soluzioni, per cui molto spesso si tratta di migrare una soluzione esistente piuttosto che progettarne una ex-novo.

## Scenario 2 – la migrazione

Negli ultimi mesi le aziende che stanno migrando o hanno migrato ad Azure sono in crescita e le problematiche sono più o meno comuni a molte realtà. In primis c’è sempre la necessità di esternalizzare lo storage solo che, a differenza di quanto detto sopra, non è più così facile intervenire nel codice e modificare il modo di salvare e reperire i files; oltretutto, quando il sistema cambia così radicalmente modo di operare, così come con l’Azure Storage, spesso si adotta uno stratagemma al limite del workaround, piuttosto che cambiare strada. Cerchiamo quindi di sviscerare un workaround cercando di dargli un aspetto più strutturato, introducendo man mano i concetti.

Il primo concetto è il concetto di Drives, ovvero di file VHD ospitati nell’Azure Storage che possano essere montati dalle istanze cloud per poter leggere dati. Si, leggere dati. Infatti la scrittura dei dati su un Drive montato può, per vincolo strutturale della tecnologia sottostante (ovvero i Blobs di Azure), essere scritto solo da un client che lo monterà (in modo esclusivo) in scrittura.

Il secondo concetto è il concetto di LocalStorage che, come sappiamo, è quello storage di natura temporanea che è parte dell’istanza nel cloud. Banalmente, è il suo spazio libero nel disco. Tuttavia il LocalStorage è uno storage che potrebbe essere resettato ad ogni riciclo dell’istanza: l’unico fattore positivo è che, traddandosi di filesystem, è molto veloce.

Il terzo concetto è relativo al File Sharing in ambiente Windows, meglio noto con il nome ddel suo protocollo, Server Message Block. La deduzione che ne dovrebbe conseguire parte da Azure come istanza di Windows Server sul Cloud, transita da Windows Server come OS dedicato ai servizi più disparati (tra cui il File Sharing) e arriva a Azure+SMB, ovvero l’abilitazione del protocollo di File Sharing sulle istanze di Azure.

Ora vediamo in dettaglio alcuni trucchi per rendere operativo lo scenario, prendendo spunto da un caso reale di cui prenderò ad esempio la topologia.

La soluzione da migrare consiste in una applicazione web che consente l’upload di file da parte dell’utente e la successiva erogazione dei file attraverso URIs cablati in strutture pre-esistenti. Durante il processo, il file in questione viene stanziato su disco in attesa di una computazione e raffinato successivamente sempre passando dal file system.

## Soluzione di base

In una siffatta soluzione è difficile intervenire univocamente dal fronte dei BLOB, per cui si adotta uno stratagemma come di seguito:

* + Worker Role “0”
    1. Si utilizza uno startup task per installare sul server virtuale i servizi di File Sharing
    2. Si crea una share di rete che punti ad un percorso su disco (es. LocalStorage)
    3. Si apre il firewall
    4. Si utilizza uno script (PS o CMD) sempre in fase di startup, per creare una utenza che abbia i permessi di lettura/scrittura sulla share di rete creata.
  + Web Role 0-N
    1. Durante lo startup si creerà una unità di rete in connessione al server 0
    2. Si imposterà il path montato nelle settings dell’applicazione in modo che i vari punti possano accedervi in lettura/scrittura.

Questa soluzione porta ad una prima considerazione: il worker role salva su disco, il che significa che al primo riciclo perderemo tutto. Quindi questo scenario è utile per una applicazione che faccia uso di dati temporanei ma non persistenti.

Se però intervenissimo tra il punto 1 e il punto 2, creando un VHD su un BLOB remoto e montandolo opportunamente (ed in scrittura) in un percorso noto, a questo punto potremmo fare puntare la share di rete a questo percorso, scavalcando di fatto il limite dell’accesso multiplo a un Windows Azure Drive, bypassandolo con un accesso SMB.

La soluzione potrebbe ancora essere migliorabile: infatti, in condizioni di High-Reliability la soluzione non garantisce uno SLA alto, per via del singolo nodo di condivisione file che potrebbe venire a mancare. Quindi, per raffinare l’idea precedente, è necessario un meccanismo, parzialmente dipendente dall’API di controllo del ciclo di vita di Azure, per l’attivazione di una seconda istanza Worker di condivisione files.

* 1. **Nota:** Questa soluzione implica un meccanismo distribuito di elezione del supernodo di condivisione e di discovery da parte dei client.

## Soluzione di alta disponibilità e affidabilità

* + Worker Role 0-N
    1. Si utilizza uno startup task per installare sul server virtuale i servizi di File Sharing
    2. Si “prova” a montare il VHD su BLOB remoto (aka Azure Drive)
       1. Se ci si riesce si è “eletti” a supernodo
       2. Se non ci si riesce ci si mette in sleep/wait e si ritenta indefinitamente
          1. **Note:** In caso il worker supernodo cada, un secondo worker riuscirebbe ad ottenere il lease sul VHD e quindi a montarlo in scrittura, diventando così il nuovo supernodo.
    3. Si crea una share di rete che punti al percorso del VHD montato
    4. Si apre il firewall
    5. Si utilizza uno script (PS o CMD) sempre in fase di startup, per creare una utenza che abbia i permessi di lettura/scrittura sulla share di rete creata.
  + Web Role 0-N
    1. Durante lo startup si cicleranno i worker role del servizio “tentando” una scrittura su una share di rete predeterminata:
       1. Quando il client riuscirà a completare la scrittura senza eccezione, avrà trovato il supernodo di condivisione.
    2. Si imposterà il path montato nelle settings dell’applicazione in modo che i vari punti possano accedervi in lettura/scrittura.

Questa soluzione conclude lo scenario impostando una strategia di maggior tutela per utilizzare uno store unico lungo le nostre istanze di Azure.

## Approfondimenti

Un approfondimento tecnico con gli esempi di codice per perseguire la soluzione sopra sono disponibili all’indirizzo: <http://bit.ly/fAfmey>.

#### di Roberto Freato ([blog](http://dotnetlombardia.org/blogs/rob/default.aspx)) - Microsoft MVP

* 1. [*Altri articoli di Roberto Freato nella Libreria*](http://sxp.microsoft.com/feeds/3.0/msdntn/TA_MSDN_ITA?contenttype=Article&author=Roberto%20Freato) [](http://sxp.microsoft.com/feeds/3.0/msdntn/TA_MSDN_ITA?contenttype=Article&author=Roberto%20Freato)