Lezione 9 modulo 2

In questo modulo parleremo di 'Server Farm'. Abbiamo già visto che le architetture dal punto di vista fisico hanno diversi TIER, quindi abbiamo parlato ad esempio di architetture a 3 TIER, supponendo di avere in genere una macchina per livello. In questo caso appunto stiamo parlando di livelli fisici quindi di macchine. Abbiamo già accennato in precedenza che in un livello fisico potremmo avere una o più macchine in realtà, quindi ad esempio situazioni in cui abbiamo più server associati ad un livello, ad esempio se abbiamo una architettura P A D, in cui abbiamo a livello di applicazione messo un livello intermedio, possiamo avere più server che si dedicano a queste applicazioni. In questo modulo andiamo a studiare cosa vuol dire avere più server associati a un certo livello fisico e innanzitutto discutiamo quali possono essere gli obiettivi di fare questo. Quello che abbiamo in generale all'interno di una macchina è un limite intrinseco computazionale di questa macchina, quindi nel caso il nostro carico sia superiore a quello che la macchina può sostenere, vorremmo poter aggiungere altre macchine in grado di sostenere complessivamente questo carico. Quindi il primo obiettivo è quello di sostenere il carico; poi un altro aspetto che si può presentare è il fatto che la macchina possa andare in manutenzione o possa avere dei problemi hardware e quindi quello che abbiamo se abbiamo un'unica macchina dedicato a un livello non siamo in grado di operare e quindi l'altro aspetto che noi vogliamo avere è quello di poter gestire un sistema in modo che sia più affidabile e in particolare ci interessa avere una certa tolleranza ai guasti. Un altro aspetto che vogliamo trattare in questo modulo è il fatto che il carico può variare nel tempo e quindi non solo crescere nel tempo ma anche eventualmente crescere poi calare e quindi avere dei picchi. Quindi un altro aspetto da gestire è la variabilità: tutti questi aspetti li andremo a esaminare per quanto riguarda la gestione appunto di server in un'organizzazione e quindi tipicamente in quei livelli che sono gestiti all'interno dell'organizzazione. Abbiamo detto quindi che la nostra esigenza è quella di poter sostenere dei carichi che aumentano nel tempo: definiamo come 'Scalabilità' la capacità di un sistema di sostenere un carico che aumenta nel tempo e questo può essere dovuto a motivi diversi: può essere dovuto al fatto che aumenta il numero degli utenti oppure che gli utenti eseguono più operazioni sul sistema; quindi ci aspettiamo di avere una crescita nel tempo di quelle che sono le attività svolte sul sistema. Quando dobbiamo sostenere questa crescita di carico, abbiamo parlato prima del fatto che vogliamo poter migliorare le prestazioni di un certo livello aggiungendo macchine, in realtà noi avremo due possibilità: quella che viene chiamata lo 'Scale Up', il caso in cui il numero dei nodi non cambia e quindi potenzio la macchina, posso aggiungere RAM, oppure CPU, oppure STORAGE, in genere questo mi porta a raggiungere un massimo possibile di capacità di elaborazione della macchina e quindi riuscirò a sostenere l' aumento di carico fino a un certo punto con le macchine disponibili. Un altro approccio è quello che viene chiamato 'SCALE OUT': in questo caso io introdurrò una ridondanza, quindi utilizzerò più macchine per svolgere le operazioni che si vogliono eseguire; quello che avverrà in questi casi è che dovrò gestire in generale più macchine che mi forniscono delle capacità di elaborazione superiori e dovevo farlo utilizzando anche una distribuzione del carico su queste macchine e quindi tipicamente ci sarà un componente aggiuntivo che verrà chiamato 'LOAD BALANCER' che andrà a distribuire il carico sulle macchine disponibili. Un'altra caratteristica di questo tipo di soluzione è che potrò utilizzare anziché macchine molto potenti per reggere un carico molto significativo, tante macchine di potenza inferiore, quindi a basso costo e ottenendo gli stessi risultati e quindi utilizzando tanti set di fascia bassa potrò fare delle operazioni che abbiamo chiamato di 'down sizing', un aspetto che diventa importante quando vado a considerare il numero, quindi i nodi variabili nel caso in cui abbiamo una ridondanza e quello del dimensionamento di questo numero, ci sono due rischi: quello del sovradimensionamento e quindi configuro il mio sistema ipotizzando un carico superiore a quello reale e quindi in questo caso avrò dei problemi di costo, oppure l'altro problema che potrei avere è il sottodimensionamento: in questo caso i numeri di nodi risultano sufficienti e sarà necessario riprogettare le architetture nuovamente. Un altro aspetto che può essere critico nel dimensionamento dei server è il fatto che il carico può essere variabile, quindi noi possiamo avere delle situazioni ad esempio di picchi che ad esempio in un sito di e-commerce possono corrispondere al fatto che ho un periodo particolare di vendita quindi ad esempio io ho delle

festività natalizie in cui abbiamo un carico sul sistema superiore a quello normale e quindi in realtà quando parliamo di dimensionamento non parliamo solo del fatto che possa aumentare il carico ma può anche diminuire e arrivare a livelli inferiori, quindi un'altra proprietà che noi vorremmo poter raggiungere è quella della elasticità, quindi la capacità di aumentare ma anche diminuire il carico a seconda delle esigenze. Andremo a vedere in questo modulo e in quelli successivi diversi tipi di architetture che possono consentire di gestire un aumento di carico in diversi modi con architetture diverse e anche poi andremo a vedere come si può trattare il problema del carico variabile con architetture più avanzate. Cominciamo a vedere come creare delle cosiddette 'Server Farm', quindi insiemi di server dedicati a un certo TIER, quindi faremo un'operazione di 'skill out' e andiamo a vedere quali possono essere gli approcci che possiamo avere nel creare architetture di questo tipo. Innanzitutto ci sono due approcci possibili: quello chiamato 'CLONING', quello chiamato 'PARTITIONING'. Nella clonazione noi sostanzialmente creeremo delle coppie uguali di un certo nodo per sostenere un carico maggiore, nel partition invece avremo più nodi al posto di un nodo e distribuiremo su questi nodi le funzionalità che vorremmo fornire in quel livello. Vedremo che potranno avere caratteristiche diverse e discuteremo soprattutto anche la condivisione o meno dei dischi che vengono utilizzati dai server. Per la clonazione noi abbiamo detto che creeremo delle coppie uguali dei nodi e abbiamo due tipi di configurazioni a seconda che le informazioni su disco vengano condivise oppure no. Quindi avremo una configurazione chiamata 'Shared-disk' e una configurazione chiamata 'Shared-nothing' quando ciascun server avrà il proprio storage che non viene condiviso con altri server. vediamo le configurazioni possibili: quindi uno 'shared-nothing', qui vediamo quello che rappresenterà per noi il load-balancer, distribuirà il carico su più server che saranno identici come funzionalità e ciascuno dotato di un proprio supporto per la memorizzazione di massa. Questo tipo di approccio sarà comodo quando devo fare delle operazioni che sono soprattutto di lettura, quindi io posso duplicare i dati nei dischi a cui vanno ad accedere le applicazioni che quindi potranno lavorare contemporaneamente; è chiaro che nel caso in cui io abbia anche degli aggiornamenti dei dati un'architettura di questo tipo può porre dei problemi perché poi dovrò sincronizzare i dati sulle varie copie quindi dovrò a livello applicativo gestirmi l'allineamento dei dati. Per evitare questo l'architettura che invece è tipica di applicazioni in cui abbiamo anche una esigenza di condividere dei dati è quella che crea uno strato in cui andiamo a condividere i dati e quindi abbiamo le applicazioni che vedono tutte gli stessi dati e quindi se viene fatto un aggiornamento da una applicazione sarà visibile anche dalle altre applicazioni. Quindi per applicazioni in cui abbiamo lettura e scrittura una configurazione tipica è quella che viene denominata 'Shared Data'. Anche in questo caso abbiamo nodi che fanno tutti le stesse operazioni, quindi dal punto di vista applicativo ho delle coppie, dei nodi semplicemente per distribuire il carico su questi nodi in modo da poter facilmente ad esempio aggiungere un nodo nel caso il carico possa aumentare e questo in una configurazione appunto di clonazione mi consentirà appunto questa scalabilità di tipo orizzontale che è caratteristica dello Scale-out. Abbiamo detto che un altro approccio che possiamo avere quando utilizziamo delle repliche ai nodi è quello del partizionamento: in questo caso sono le applicazioni che vengono allocate su nodi diversi, quindi mentre nella clonazione supponiamo di avere tre applicazioni metto tutte e tre le applicazioni su ciascuno dei nodi clonati, nel caso di partizionamento potrò decidere di allocare solo alcune delle applicazioni a un certo nodo; quindi supponiamo ad esempio di avere tre server, potremmo avere l'applicazione 1 sul primo, la 2 sul secondo è la terza sul terzo, quindi andiamo a partizionare appunto quello che è il livello applicativo sulla base delle applicazioni. Qual è il vantaggio di fare un’architettura di questo tipo? Possiamo ottenere nel caso di malfunzionamento quello che viene chiamata la 'Graceful Degradation': ad esempio se un nodo diventa non disponibile perché devo fare delle operazioni di manutenzione, le funzionalità associate a quel nodo non saranno disponibili ma tutte le altre continueranno ad essere disponibili e quindi l'utente potrà utilizzare il sistema sia pure in modo parziale. Quindi avremo un’architettura di questo tipo: sempre un load-balancer che distribuirà le richieste di funzionalità alle varie applicazioni in cui abbiamo partizionato sul nostro sistema e supponiamo di avere appunto su ogni server applicazioni diverse. In questo caso noi vediamo che abbiamo la possibilità di aumentare i nodi dividendo applicazioni e questi nodi possono gestirsi i propri dati. Ovviamente non abbiamo risolto il problema che avevamo visto prima: cosa succede

se aumenta molto il carico di un'applicazione oppure dal punto di vista della tolleranza e guasti oppure della possibilità di avere un servizio nel caso ci sia una manutenzione noi abbiamo sempre la criticità che abbiamo ad esempio un'applicazione 4 che viene allocata solo a un nodo. Ovviamente anche nel caso del partizionamento possiamo applicare oltre al partizionamento anche la clonazione e quindi possiamo creare più nodi che corrispondono alla stessa applicazione. Questo ci porta a un'altra architettura che viene chiamata 'RAPS', come 'Reliable array of partitioned services'; quindi noi abbiamo la possibilità di avere sempre le varie partizioni, ma abbiamo usato anche più copie quindi possiamo avere più nodi uguali per la stessa partizione aumentando quindi la disponibilità del sistema potendo aumentare il carico e quindi con tutti i vantaggi della clonazione. Anche qui in questo caso ci sarà un load-balancer che avrà delle politiche di allocazione del carico non solo in base alle funzionalità che vengono richieste ma anche in base al carico delle singole macchine in modo da distribuire il carico per avere un sistema performante. Abbiamo visto quindi come creare delle server farm utilizzando clonazione e partizionamento. Vediamo un esempio di applicazione su un'architettura tipica che abbiamo già discusso, quella di applicazioni basate su web. Un'architettura a cinque livelli, in cui abbiamo un livello dati, vediamo che qua abbiamo il livello dati rappresentato da un unico server che consente di gestire i dati in modo condiviso e questo è tipico della applicazioni in cui vogliamo anche andare a scrivere i dati come abbiamo visto perché il problema è quello della gestione di dati replicati nel caso in cui il livello dati venga replicato. Quindi dove vediamo in un'architettura di questo tipo una replicazione dei nodi? La vediamo a livello di application server per cui possiamo avere più nodi che gestiscono le applicazioni in modo abbiamo visto che può essere partizionato o partizionato e clonato e poi avremo diversi livelli per lo 'script engine' e 'web server'. Ad esempio per il web server, potremmo avere tanti web server tutti uguali e quindi avremo una vera e propria clonazione e potrò gestire carichi di lavoro che aumentano semplicemente aggiungendo dei moduli di gestione di richieste http. Un altro aspetto che potrei avere nell'ambito web è il fatto che alcune richieste non siano solo le normali richieste di, basati su protocollo http ma si vogliano avere delle richieste basate su un protocollo di tipo https e quindi vogliamo avere delle funzionalità di sicurezza, quindi di cifratura dei dati, quindi gestione di riconoscimento del server oppure del client. In questo caso quindi abbiamo un caso in cui di fatto abbiamo un partizionamento: quindi vogliamo avere una gestione di alcuni server che si occupano di collegamenti https ed altri che si occupano di collegamenti http, anche per gli 'script engine' possiamo avere tanti 'script engine' in grado di produrre pagine e questo può essere fatto utilizzando ovviamente funzionalità degli altri layers, in particolare quelle applicative, ma può essere fatto da nodi che sono dedicati a questo compito e possono essere associati ai vari server oppure essere dimensionati diversamente.