Cabling & Wireless DIGITAL NEWS
Settembre 2020

Servizi distribuiti nei Data Center

[ab]Mario Baldi, [cd]Marcello Maggiora

1984 Massachusetts, il direttore e fondatore dei Media Lab del *Massachusetts Institute of Technology* (MIT), Nicholas Negroponte, durante una conferenza, sostenne che si poteva utilizzare il dito come puntatore al posto dell'ormai classico mouse mostrando in anteprima il funzionamento di uno schermo touch. Il primo iPhone, che ha portato la tecnologia touch a cambiare radicalmente il mondo di utilizzare i dispositivi, risale al 2007.
2004 Silicon Valley, California. Brice Clark, all'epoca responsabile worldwide della strategia del settore networking di Hewlett Packard, durante un Vision Talk sostenne che gli elaboratori sarebbero diventati una specie di 'connettori' di risorse poste nella rete, con sistemi operativi e funzionalità pensate per essere utilizzate immersi nella rete. Un modello questo oggi reale, pensiamo al mondo applicativo in cloud e agli elaboratori nati per il cloud come il Chromebook.
Gli esempi in questo senso potrebbero essere molti, le idee e le tecnologie che nascono nei centri di ricerca e nelle università a volte impiegano molto tempo per conquistare la consistenza necessaria e incidere significativamente nelle attività e nelle abitudini delle persone.

Oggi siamo pronti perciò a vedere decollare idee e soluzioni che da una decina di anni sollecitano l'interesse di molti osservatori e hanno raggiunto un livello di maturità tale da essere pronte per atterrare sul tavolo dei responsabili IT. Dai dati che possiamo osservare emergono due tendenze che hanno già conquistato un certo spazio e che potrebbero diventare il modello di riferimento.

La prima riguarda *l'intelligenza* delle infrastrutture ICT che sta lentamente, ma inesorabilmente, spostandosi dall'hardware al software. Un processo questo nato con la virtualizzazione, la cui diffusione ed evoluzione a poco a poco ha mostrato quanto fosse più rapido e *facile* modificare parti di infrastruttura agendo su una configurazione software piuttosto che mettendo mano direttamente ad apparati e sistemi dell'infrastruttura fisica.

Oggi infatti le varie declinazioni di *software defined* toccano praticamente ogni componente infrastrutturale, dalla rete al data center. Il futuro vedrà una grande diffusione di queste soluzioni e sarà sempre più fondato sul software. Questo approccio porterà più flessibilità, più rapidità e forse anche qualche nuova preoccupazione.
La seconda tendenza riguarda il traffico IP che in generale sta crescendo costantemente in modo significativo. Il traffico IP, secondo le Equinix Global Interconnection Index 2019, nel 2018 era di 2.500 Tbps globali e crescerà fino a oltre 13.000 Tbps nel 2022. Va sottolineato che queste proiezioni sono del 2019 e quindi non considerano l'effetto COVID-19 che ha sicuramente accelerato, probabilmente in modo irreversibile, tale crescita.
Poiché vogliamo approfondire il legame tra traffico IP e le soluzioni Software Defined, osserviamo come è composto il traffico all'interno del data center.

Emerge che quasi l'80% del traffico IP è ascrivibile alla direzione est-ovest, cioè al traffico interno al data center, seguito a distanza dal traffico "data center-utente" 14% e infine il traffico "data center-data center" circa il 9% (vedi **Fig. 1**). Al di là delle percentuali specifiche è un dato condiviso che la maggior parte del traffico IP non esce dal data center e questo pone delle problematiche non marginali nell'implementazione di servizi distribuiti basati su soluzioni software.

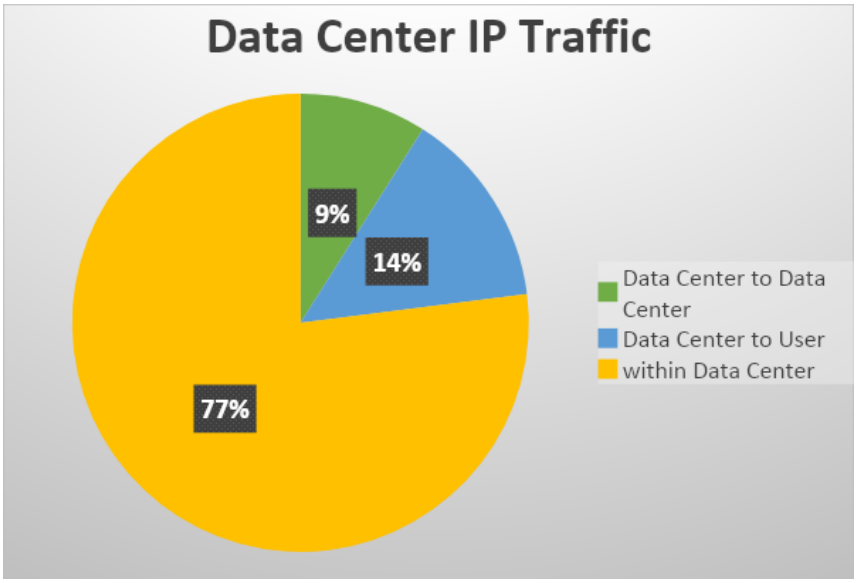

Fig. 1 - Data Center, composizione del traffico IP (fonte Cisco Sytems)

Lo scopo ultimo di un data center è quello di ospitare calcolatori che eseguono applicazioni fruibili dagli utenti finali e che producono guadagni. Tuttavia, il data center deve anche offrire funzionalità essenziali alla fruizione di tali applicazioni, tra cui la connettività di rete, servizi aggiunti di rete (quali NAT o network address translation, bilanciamento di traffico), servizi legati alla sicurezza (quali filtraggio o firewalling, identificazione e prevenzione delle intrusioni), servizi legati al salvataggio su disco (dischi condivisi e disaggregazione della memoria di massa), e servizi legati alla visibilità (raccolta di informazioni di telemetria, cattura e analisi di traffico).

Tradizionalmente i servizi sopra elencati vengono erogati da *"appliance" centralizzate* posizionate all'interno della rete del data center, eventualmente eseguite dagli apparati di rete stessi. Ovviamente, è fondamentale che il traffico che deve ricevere un certo servizio venga fatto transitare attraverso la appliance che lo eroga. Questo è relativamente semplice per servizi da applicare al traffico cosiddetto nord-sud[1], cioè che viene scambiato con Internet in ingresso ed uscita perché, come rappresentato graficamente in Fig. 2, questo transita per un punto di passaggio obbligato che è il collegamento con l'esterno.

La questione è invece più complicata per il traffico interno al data center, colloquialmente indicato come est-ovest. Infatti la rete dei data center moderni viene progettata usando la topologia Clos che offre una moltitudine di percorsi identici tra una qualsiasi coppia di stazioni non collegate allo stesso nodo di rete. Tali percorsi vengono sfruttati per aumentare la capacità di trasferire traffico tra gli host distribuendo il traffico in modo il più possibile uniforme su ognuno di essi con la tecnica nota come ECMP (Equal Cost Multi-Path) routing.

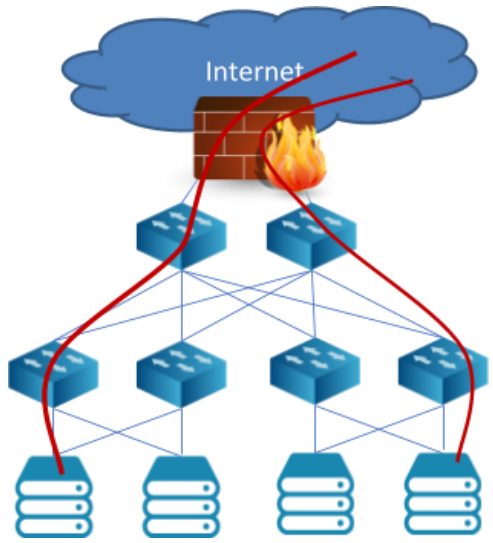

Fig. 2 – Schema con topologia Clos e flusso del traffico Nord-Sud

1 Il nome deriva dal modo in cui viene normalmente disegnata la topologia della rete di un data center (si vedano Fig. 2 e Fig. 3) con i calcolatori in basso, cioè a sud, e il collegamento a Internet in alto, cioè a nord.

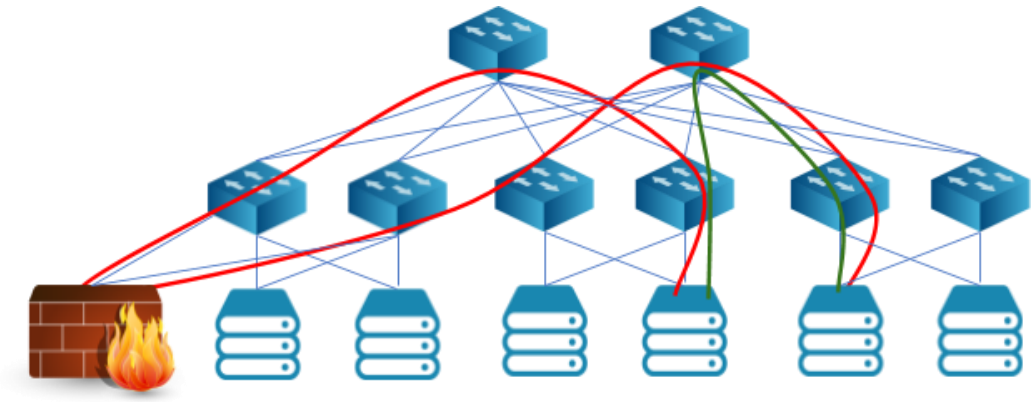

Fig. 3 – Traffic steering, invece che seguire il percorso ottimale verde, il traffico viene forzato verso il percorso rosso

Dal punto di vista dell'erogazione di servizi, questo significa che non esiste un unico punto attraverso cui passa il traffico dove posizionare le appliance. E non si vuole certo progettare la rete in modo da creare un tale punto di passaggio perché, vista l'elevata quantità di traffico est-ovest, esso costituirebbe uno strozzamento della rete (comunemente indicato come bottleneck o collo di bottiglia) che porrebbe un limite alle prestazioni. Dunque, l'unica possibilità è quella di alterare il normale percorso del traffico per farlo passare attraverso le appliance che lo debbono elaborare, dovunque queste siano collegate nella rete (operazione comunemente chiamata traffic steering o traffic stitching), come mostrato in Fig. 3. Invece che seguire il percorso ottimale verde, il traffico viene "dirottato" verso l'appliance lungo il percorso rosso. Questo ha implicazioni sia sulla complessità dell'architettura protocollare di rete, sia sulle prestazioni.

- Esistono due opzioni comunemente utilizzate per realizzare il traffic steering verso appliance (si veda [**1**] per una spiegazione più dettagliata):
  - l'uso di router che supportano tabelle di routing multiple, una tecnologia comunemente indicata come *VRF* o Virtual Routing and Forwarding;
  - la creazione di vari domini di livello 2 in un *overlay* al di sopra della interconnettività di livello 3 fornita dai nodi della rete Clos.

  Entrambe rappresentano una complessità aggiuntiva nella configurazione, gestione e ricerca guasti della rete.

- D'altro canto, su una topologia Clos far prima raggiungere al traffico l'appliance e poi farlo andare verso un calcolatore implica attraversare due volte gli apparati di dorsale (spine) e l'interconnessione tra essi e le foglie (leaf), creando quello che è comunemente noto come *effetto trombone* (perché il percorso nella rete ricorda la forma dello strumento musicale, come si vede in Fig. 3).
  - Come chiaramente visibile confrontando il percorso rosso con quello verde in Fig 3, questo implica duplicare (o *moltiplicare* per N quando si debbano attraversare N appliance) il traffico sulla rete, cosa che è particolarmente critica considerando i volumi estremamente elevati di traffico est-ovest.
  - Dovendo attraversare più apparati di rete, il percorso rosso è soggetto ad un *aumento di latenza e jitter* (variazione della latenza) rispetto a quello verde. Questo è particolarmente critico per la comunicazione tra i vari stadi delle applicazioni web che si aspettano latenze molto basse e deterministiche e che praticamente generano la totalità del traffico est-ovest.

In aggiunta a tutti questi svantaggi, per molti dei servizi che si vogliono offrire è praticamente impossibile (o estremamente costo) realizzare appliance con le tecnologie attualmente disponibili che siano in grado di sostenere i volumi tipici del traffico est-ovest. Per ovviare a questo problema si può pensare di avere più istanze di ogni appliance e distribuire il traffico tra di esse, ma questo complica ulteriormente il routing nella rete ed ha implicazioni sulla condivisione dello stato tra le varie istanze di appliance che realizzano servizi che richiedono memoria storica.

L'unica soluzione per superare tutti questi svantaggi è quella di erogare i servizi in modo *distribuito* (invece che tramite appliance discrete) in modo che ogni pacchetto incontri il servizio su qualsiasi percorso esso segua verso la propria destinazione in base al normale routing.

Questo può essere fatto in due modi differenti, che possono eventualmente essere combinati.

- *Nodi di rete*: creando un servizio distribuito in tutti i nodi di accesso (leaf) si ha la certezza che qualsiasi pacchetto entri nella rete possa ricevere il servizio.
- *Calcolatori*: realizzando un servizio in modo distribuito all'interno di tutti i calcolatori nel data center il traffico può ricevere il servizio quando viene generato o ricevuto.

Le due soluzioni hanno molti punti in comune e la principale differenza è il livello di distribuzione. Nel caso si utilizzino nodi di rete il livello di distribuzione è minore che se si utilizzano i calcolatori. Un maggiore livello di distribuzione corrisponde ad una maggiore complessità di gestione, ma d'altro canto ad una migliore capacità di sopportare la crescita di traffico dovuta ad un aumento della dimensione del data center. In particolare, se un servizio viene realizzato all'interno dei calcolatori, aumentando il numero incrementa proporzionalmente il livello di distribuzione del servizio, quindi la quantità di risorse disponibili per l'esecuzione del servizio.

Va però tenuto presente che il ruolo principale sia dei calcolatori, sia dei nodi di rete non è quello dell'erogazione dei servizi, ma rispettivamente l'esecuzione delle applicazioni per cui il data center è stato realizzato e la fornitura della necessaria connettività. Dunque è bene che le risorse disponibili in termini di memoria e processore vengano dedicate a tali funzionalità, mentre idealmente i servizi possono essere erogati da hardware specializzato installato sui dispositivi.

Questo è per l'appunto l'approccio seguito da Pensando Systems che ha progettato una soluzione completa per realizzare e gestire servizi distribuiti che include un circuito integrato appositamente per l'esecuzione di servizi (ASIC - *application specific integrated circuit*), il software da eseguire su di esso e un sistema di gestione centralizzato. La prima linea di prodotto basata su questo circuito integrato è una scheda per calcolatore chiamata Distributed Services Card (DSC). Questa scheda ha un'interfaccia PCIe e due interfacce Ethernet, quindi può essere utilizzata in un calcolatore al posto di una normale scheda Ethernet. Però siccome l'ASIC include sia processori generici (core ARM), sia processori specializzati nell'elaborazione di pacchetti programmabili tramite il linguaggio P4 (*Programming Protocol-independent Packet Processors*) [**3**,**4**], può essere programmato per realizzare servizi aggiuntivi che si vengono così a trovare sul percorso dei pacchetti in ingresso e in uscita dai server. L'architettura [**2**] è progettata in modo da garantire l'esecuzione di funzionalità complesse alla velocità di linea delle due interfacce Ethernet così che l'inserzione dei servizi non causi un abbassamento delle prestazioni. Visto che l'esecuzione di questi servizi avviene esclusivamente sulla scheda, le risorse del calcolatore non vengono coinvolte e possono essere completamente dedicate alle applicazioni "paganti".

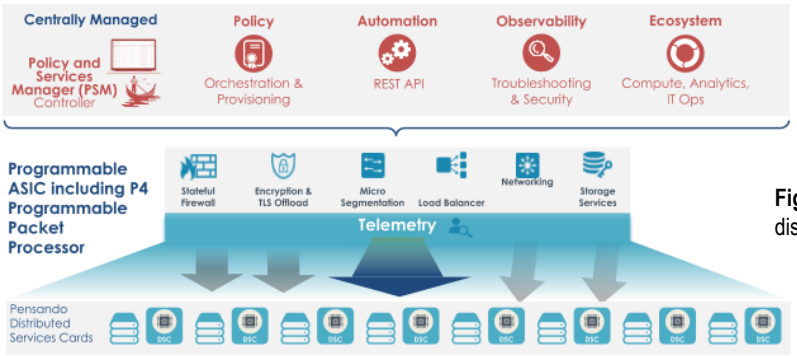

Fig. 4 - La piattaforma di Pensando per la distribuzione dei servizi alla periferia dei data center

L'unica problematica tra quelle elencate in precedenza che rimane aperta è la complessità del configurare servizi con un elevato livello di distribuzione e mantenere sotto controllo l'esecuzione. Nella piattaforma di Pensando questo è risolto dal Policy and Services Manager (PSM o gestore di politiche e servizi) che offre un unico punto di configurazione e controllo dei servizi per l'utente che può esprimere quello che i servizi erogati dalle varie DSC devono ottenere. Il PSM seguendo un paradigma *intent-based* si occupa in modo indipendente di configurare opportunamente ognuna delle DSC coinvolte. Se nuove DSC vengono attivate, sono automaticamente configurate in modo coerente con le altre così che i servizi vengano erogati anche da queste.

In sintesi, come mostrato nella figura 4, le politiche di rete, sicurezza e storage sono specificate sulla PSM come se i servizi corrispondenti fossero erogati in modo centralizzato, senza cioè preoccuparsi del fatto che tali politiche siano imposte da un elevato numero di DSC alla frontiera tra la rete e i calcolatori, in modo altamente distribuito. Questo garantisce alla piattaforma la stessa capacità di espansione del data center stesso: più calcolatori si installano, più risorse diventano disponibili non solo per l'esecuzione delle applicazioni, ma anche per i servizi. Nello stesso tempo la PSM raccoglie informazioni di telemetria molto ricche dalle DSC, sia sullo stato delle risorse delle schede, sia sul traffico di rete, e fornisce all'utente una visione aggiornata e globale dello stato della piattaforma, dei servizi e della rete.

Il posizionamento della DSC nei calcolatori e l'architettura dell'AIC su cui è basata la rendono adatta a svariati casi d'uso, come per esempio:

- la presa in carico (*off-load*) di funzionalità tipicamente realizzate (con significativo impegno di risorse di calcolo) del processore dei calcolatori, come la (decifratura e/o (decompressione dei dati, la segmentazione e il riassemblaggio dei messaggi TCP, la terminazione delle sessioni TLS (Transport Layer Security);
- la virtualizzazione di dischi remoti tramite soluzioni quali NVMe-oF (Non Volatile Memory express over Fabric);
- la microsegmentazione, cioè la possibilità di introdurre politiche di sicurezza che limitano la connettività a livello di singolo "workload" nel data center, sia esso stazione, o macchina virtuale, o container, senza richiedere la redirezione del traffico che risulta nell'effetto trombone e gli svantaggi che ne derivano. ✓

[a] *Pensando Systems, Inc., 570 Alder Dr., Milpitas, CA 95035, USA*
[b] *Dipartimento di Automatica e Informatica, Politecnico di Torino, c.so Duca degli Abruzzi 24, 10129 Torino*
[c] *Compagnia di San Paolo Sistema Torino, piazza Lorenzo Bernini 5 - 10138 Torino*
[d] *CNR-IEIIT, Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni, c.so Duca degli Abruzzi 24, 10129 Torino*
baldi@pensando.io - marcello.maggiora@ieiit.cnr.it

Riferimenti bibliografici

[1] M. Baldi, S. Gai, "Traffic Tromboning: Why, What, Where, How, and How Not," blog post, https://blog.baldi.info/trombone, June 2020.

[2] M. Baldi, D. Crupnicoff, S. Gai, "Programmable Data Plane Architecture for Distributed Services at the Network Edge," 7th IEEE International Conference on Software Defined Systems (SDS2020), Paris, France. July 2020.

[3] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. Programming protocol-independent packet processors. In ACM SIGCOMM Computer Communications Review (CCR), volume 44, July 2014.

[4] P4.org. P416 language specification. https://github.com/p4lang/p4-spec/tree/master/p4-16/spec, May 2017.