Project: Optimizing Hardware usage in multi-cloud environment

Teacher: Henry Muccini
Project Partner: Daniele Spinosi, Micron Technology

Deliverable Template

Date	**13/01/2020**
Team ID	**qwertyasdf**

Team Members		
Name and Surname	**Matriculation number**	**E-mail address**
Leonardo Serilli	274426	leonardo.serilli@student.univaq.it
Gabriele Colapelle	274560	gabriele.colapelle@student.univaq.it
Alessandro D’Orazio	275328	alessandro.dorazio2@student.univaq.it
Pietro Ciammaricone	274427	pietro.ciammaricone@student.univaq.it

Table of Contents

Challenges e analisi dei rischi

Risk	Date the risk is identified	Date the risk is resolved	Explanation on how the risk has been managed
T1 downtime I servizi T1 devono avere il **99.999% di disponibilità**	**5/12/2020**	**7/12/2020**	È stata svolta un’**analisi sul minimo numero di server locali richiesto per gestire la totalità dei servizi T1**, arrivando a un massimo crash di 45 server, poiché i servizi **T1 sono gestibili dai 15 rimanenti**; sono quindi sempre disponibili fino al limite di crash tollerato
Gestione Fault Tolerance	**9/12/2020**	**10/12/2020**	**La fault tolerance** è gestita principalmente a livello hardware, per via dell’utilizzo di ogni cpu al **50%** in modo che un server possa **gestire l’eventuale fallimento** di un altro, inoltre superato il limite di fault tolerance, è prevista la gestione della totalità dei servizi T2 da parte del cloud pubblico, nonostante la riduzione delle prestazioni.
Bilanciamento Workload CPU	**9/12/2020**	**9/01/2021**	Imponendo l’**utilizzo medio delle cpu di ogni server al 50%** e facendo un’**analisi statistica**, abbiamo ottimizzato il **numero di servizi T1 e T2 gestibili dall’hardware locale. Assumendo che il 40% di servizi allocati a siano di tipo T1, ogni factory** dovrà gestirne **160** e, avendo a disposizione **288 CPU** per factory, possiamo allocare anche **124 servizi T2,** mantenendo un CPU workload medio del 50%. Inoltre, in caso del crash di più della metà della struttura il workload dei servizi T2 locali può essere degradato fino a un massimo del 20%, in modo tale da poter gestire in una sola factory tutti i servizi allocati localmente.

Monitoraggio Server	**1/01/21**	**12/01/21**	Lo **stato dei server deve essere continuamente tenuto sotto controllo, insieme allo stato corrente dei servizi in esecuzione. N**el caso in cui uno di questi diventi non disponibile, il servizio deve essere a**ssegnato a un altro server, possibilmente dal punto in cui è stato interrotto.** Inoltre il monitoraggio non deve essere limitato a una sola factory, ma **ognuna di queste deve poter accedere allo stato di tutta la struttura**, al fine di permettere che i servizi di una factory possano essere spostati su un’altra.

Raffinamento delle specifiche

1. **I servizi si dividono in due fasce di criticità**, T1 e T2:
    a. **Servizi T1**: servizi safety critical, business critical e mission critical;
    b. **Servizi T2**: servizi che non impattano sulla produzione, ad esempio quelli che collezionano statistiche di produzione .

2. Ogni servizio/applicazione si deve occupare di **una funzionalità.** Questo implica che ogni servizio deve svolgere solamente le operazioni che riguardano l’ambito di quella funzionalità.

3. I servizi T1 devono avere il **99.999% di disponibilità**; ogni servizio identificato come T1 deve essere eseguito all’interno della fabbrica e non deve mai essere inattivo, poiché ne risentirebbe la produzione.

4. La fault tolerance deve essere garantita principalmente a livello hardware.

5. I data center interni alla factory devono ospitare i servizi T1 e inoltre devono anche essere collegati tramite un servizio LAN dedicato a bassa latenza.

6. I servizi T2 possono essere eseguiti sia sul **cloud privato** che sul **cloud pubblico**.
    a. Se il cloud privato ha una capacità computazionale sufficiente ad eseguire dei servizi T2, garantendo comunque la fault tolerance, allora li eseguirà
    b. Tutti gli altri servizi T2 saranno eseguiti sul cloud pubblico

7. Un servizio T2 può essere inattivo per un massimo di due ore senza impattare sulla produzione, quindi Il contratto WAN ed i Cloud Provider per tali servizi deve garantire un MTTR minore di due ore.

8. In caso di “grandi fallimenti”, i servizi **T2** devono poter operare anche con **prestazioni degradate.**

9. Tra data center dislocati in factory ci deve essere una **WAN privata** a bassa latenza

10. I **servizi MES** devono essere inseriti in container Docker e devono essere istanziati in un **cloud privato** OpenShift**.**
    a. I servizi devono essere scalati e bilanciati con **Kubernetes**.

11. L’utilizzo della **CPU** deve essere **massimizzato**
    a. Ogni server ha un **50% di utilizzo di CPU medio**
    b. I servizi **T1** lavorano con il **40%** di cpu workload totale
    c. I servizi **T2** lavorano con il **60%** di cpu workload totale

12. Una Struttura ha **3 Factory**

13. Ogni **Factory** ha 2 **data center**

14. Ogni data center ha **10 server di cui un master node server**
    a. I server possono avere più di una cpu, al fine di bilanciare in modo corretto il workload totale, che deve restare sotto al 50% e arrivare all 100% in caso di grandi fallimenti

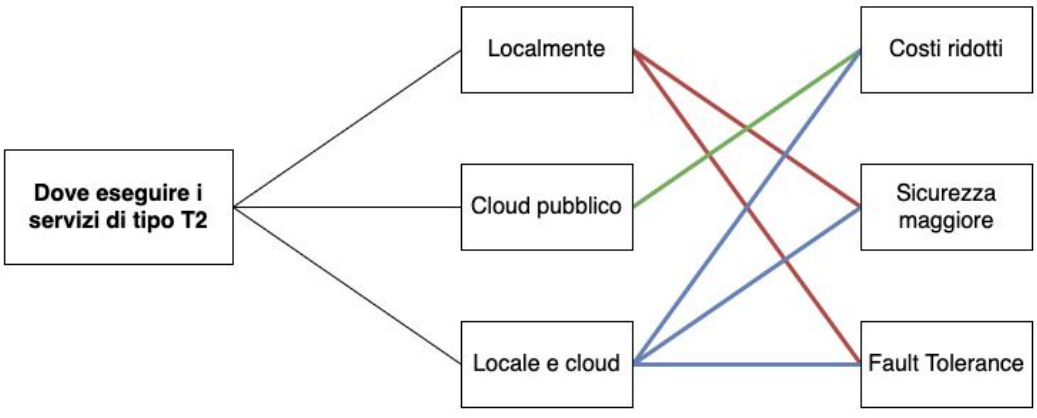
15. Nel caso di **crash** di un data center, deve essere garantita abbastanza potenza di calcolo per continuare a gestire tutti quanti i servizi richiesti

Decisioni di design

Esecuzione dei servizi di tipo T2

I servizi di tipo T2 vengono eseguiti sia in locale, sia in cloud. È stata presa questa decisione poiché in questo modo vengono ridotti i costi eseguendo dei servizi sul cloud, ma allo stesso tempo abbiamo notato che è presente abbastanza potenza di calcolo per eseguire anche parte dei servizi T2 all’interno della fabbrica.

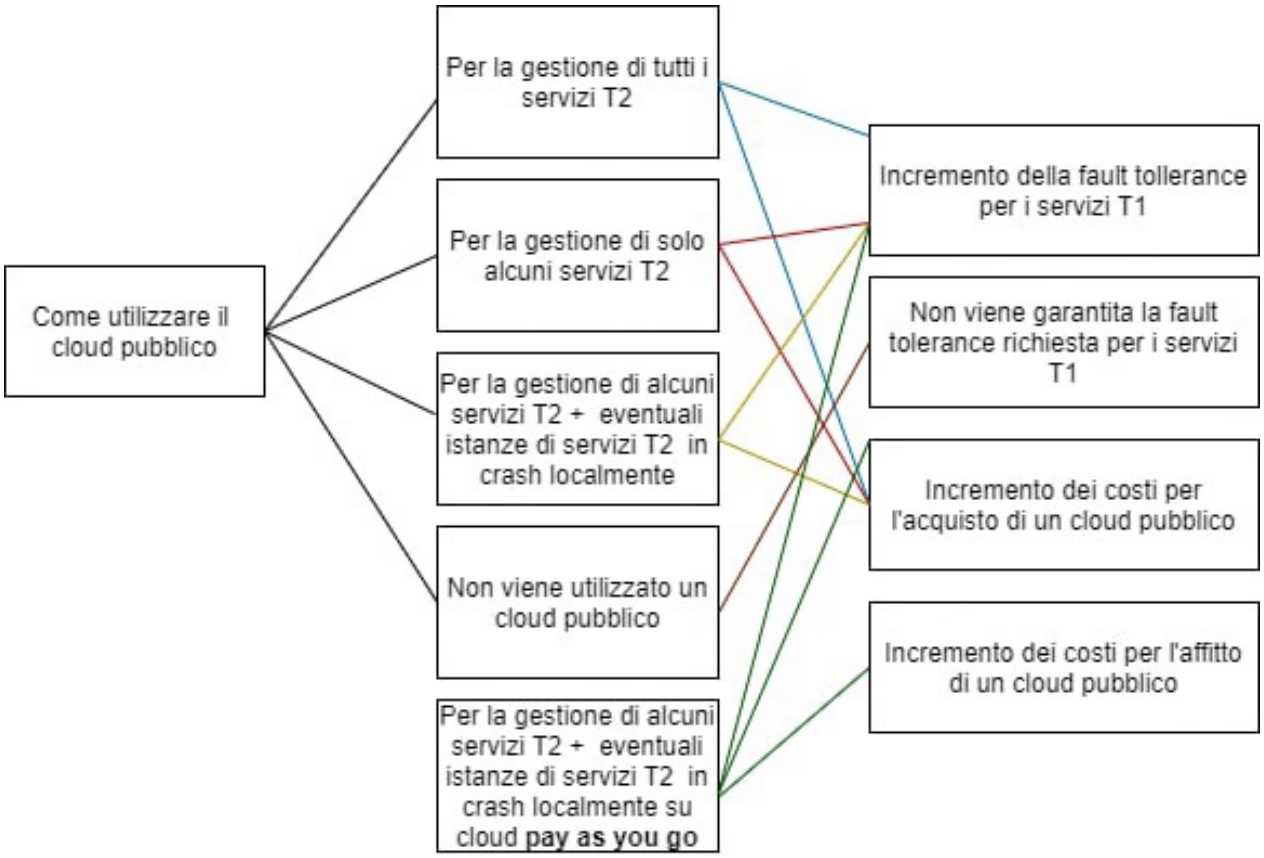
Concern		Dove eseguire i servizi di tipo T2
Alternative(s)		1. Esclusivamente in locale 2. Esclusivamente sul cloud 3. Sia in locale che sul cloud
Ranking criteria		1. Costi per l’acquisto dell’hardware 2. Sicurezza relativa alla gestione dei dati 3. Fault tolerance in caso di guasti
Architectural decision	**Identifier**	6
	Description	I servizi T2 verranno eseguiti sia localmente che sul cloud pubblico, al fine di trovare un compromesso tra fault tolerance, sicurezza e costi
	Rationale	I servizi T2 vengono eseguiti sia localmente che su cloud



Cloud pubblico

Utilizzo di un cloud pubblico per la gestione di parte dei servizi T2, ed utilizzo dello stesso con prestazioni ridotte in caso di crash di data center interni che ospitano servizi T2.

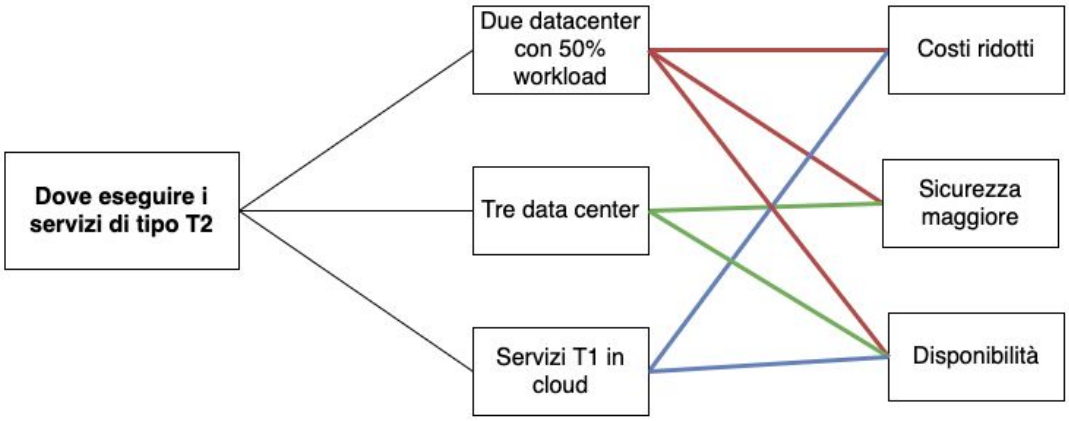
Concern		Utilizzo del cloud pubblico
Alternative(s)		1. Gestione di tutti i servizi T2 2. Gestione di parte dei servizi T2. 3. Gestione di parte dei servizi T2 + istanze di servizi T2 crashati localmente 4. Gestione di parte dei servizi T2 su un cloud pay as you go 5. Non viene utilizzato un cloud pubblico
Ranking criteria		1. Costi relativi all’acquisto di un cloud pubblico. 2. Costi relativi all'affitto di un cloud pubblico (pay as you go). 3. Costo relativo all’hardware e la sua manutenzione in caso non si volesse utilizzare un cloud pubblico
Architectural decision	**Identifier**	6b
	Description	Utilizzo di un cloud pubblico per la gestione di alcuni dei servizi T2, ed utilizzo dello stesso con prestazioni ridotte per continuare a gestire istanze di servizi T2 che non possono più essere gestite localmente a causa di fallimenti all’interno delle factory
	Rationale	Gestione di servizi T2 su cloud pubblico per garantire un sufficiente hardware interno per la gestione dei servizi T1, anche in caso di grandi fallimenti.


Come utilizzare il cloud pubblico
Per la gestione di tutti i servizi T2
Per la gestione di solo alcuni servizi T2
Per la gestione di alcuni servizi T2 + eventuali istanze di servizi T2 in crash localmente
Non viene utilizzato un cloud pubblico
Per la gestione di alcuni servizi T2 + eventuali istanze di servizi T2 in crash localmente su cloud pay as you go
Incremento della fault tollerance per i servizi T1
Non viene garantita la fault tolerance richiesta per i servizi T1
Incremento dei costi per l'acquisto di un cloud pubblico
Incremento dei costi per l'affitto di un cloud pubblico

Fault Tolerance

La Fault tolerance viene garantita principalmente a livello hardware, prevedendo che ogni factory abbia 2 data center; ogni CPU ha un carico computazionale del 50%, in modo tale che nel caso in cui un data center smetta di essere operativo, l’altro può eseguire i servizi di entrambi i data center. Inoltre è possibile degradare le prestazioni dei servizi T2 locali fino a un massimo del 20% in modo da poter gestire tutti i servi T1 e T2 locali in una sola factory.

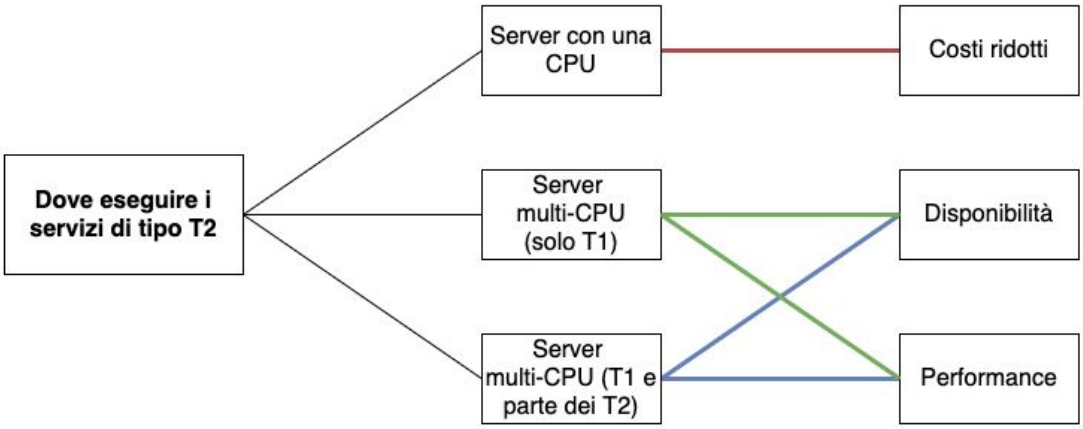
Concern		Garantire operabilità dei servizi T1 al 99.999%
Alternative(s)		1. Due datacenter con il 50% del carico 2. Tre data center (il terzo in funzione solo quando uno dei due “principali” smetta di funzionare” 3. Esecuzione dei servizi T1 in cloud qualora si verifichino malfunzionamenti
Ranking criteria		1. Sicurezza. Dei malintenzionati non devono essere in grado di manomettere i servizi T1. 2. Costo. Bisognerebbe evitare il più possibile l’acquisto di hardware che sarà inutilizzato. 3. Disponibilità.
Architectural decision	**Identifier**	4
	Description	I data center opereranno sotto il 50% di carico in modo tale da garantire, nel caso in cui uno fallisca, che l’altro possa prendersi in carico il workload del data center che ha smesso di funzionare. Inoltre le prestazioni dei T2 possono essere degradate fino al 20% dei workload, per poter gestire tutti i servizi assegnati localmente in una sola factory
	Rationale	Due data center con 50% di workload


Dove eseguire i servizi di tipo T2
Due datacenter con 50% workload
Tre data center
Servizi T1 in cloud
Costi ridotti
Sicurezza maggiore
Disponibilità

Bilanciamento del workload

Ogni server della factory è multi-cpu, in modo tale che ogni factory abbia un 50% di workload medio, suddiviso per 160 servizi T1 e 128 T2. Questa decisione è stata presa per avere dei server che possano eseguire vari servizi in parallelo. Infatti una cpu è in grado di gestire due servizi T1 (40% cpu workload ognuno) o un servizio T1 e uno T2 (40% + 60% di cpu workload), ma non due servizi T2 (arriverebbe al 120% di cpu workload). Inoltre le prestazioni dei T2 possono essere degradate al 30% o al 20% di workload, così che una cpu possa gestire o 2 T1 e un T2 (al 20%) o 1 T1 e 2 T2 (al 30%).

Concern		Come bilanciare il workload totale della CPU
Alternative(s)		1. Server con una CPU 2. Server multi-CPU, dedicata ai servizi T1 3. Server multi-CPU, per servizi T1 e parte dei servizi T2
Ranking criteria		1. Costi relativi all’acquisto dell’hardware 2. Disponibilità. I servizi T1 devono avere il 99,999% di disponibilità 3. Performance. Il workload della CPU non deve essere troppo alto
Architectural decision	**Identifier**	11
	Description	I server sono multi-CPU, ed eseguono sia i servizi T1, che, in base al loro workload, parte dei servizi T2.
	Rationale	I server sono multi-CPU ed eseguono sia servizi T1 che (parte dei) servizi T2

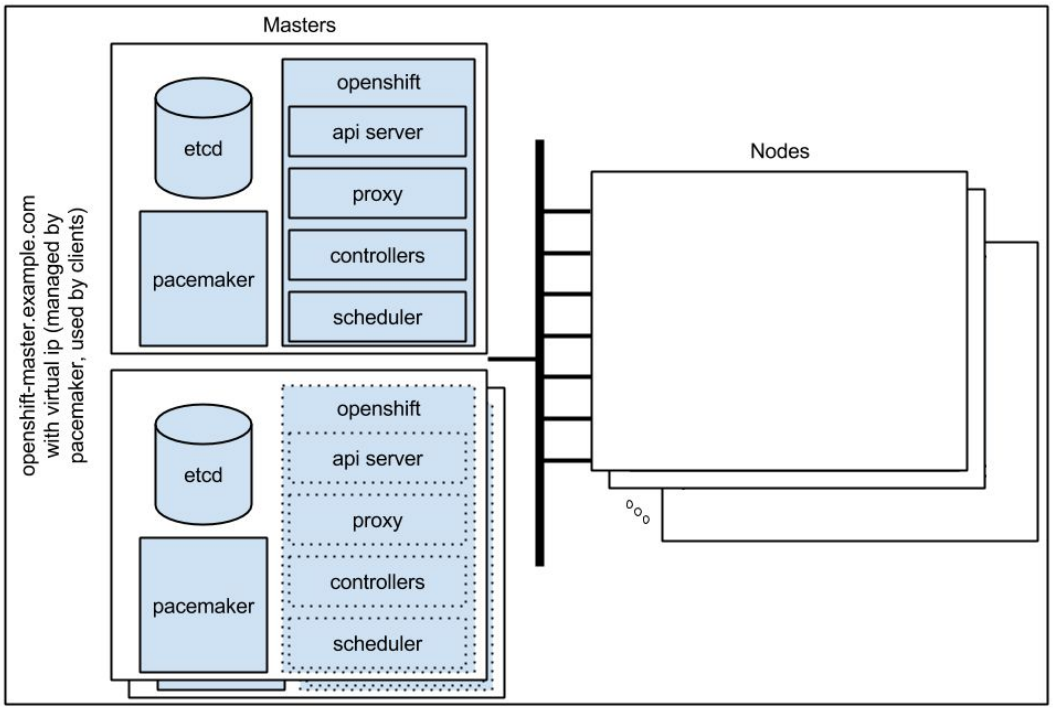

Dove eseguire i servizi di tipo T2
Server con una CPU
Server multi-CPU (solo T1)
Server multi-CPU (T1 e parte dei T2)
Costi ridotti
Disponibilità
Performance

Master Node

L'architettura dei datacenter descritta fin'ora è gestita da OpenShift, quest'ultimo prevede l'utilizzo di un Master host (o server) il quale gestisce le più importanti componenti di controllo e assegnazione dei task all'interno del datacenter.
Tra i compiti del master host sono incluse: la gestione delle API di kubernetes (configurazione dei pods, assegnazione dei pods ai nodi, sincronizzazione tra pods), etcd (store permanente di dati chiave-valore che mantiene lo stato del master in caso esso debba essere spostato o riavviato),
controller manager (controlla i cambiamenti in etcd e replica tali cambiamente sugli altri host della rete mediante le API di kubernetes).

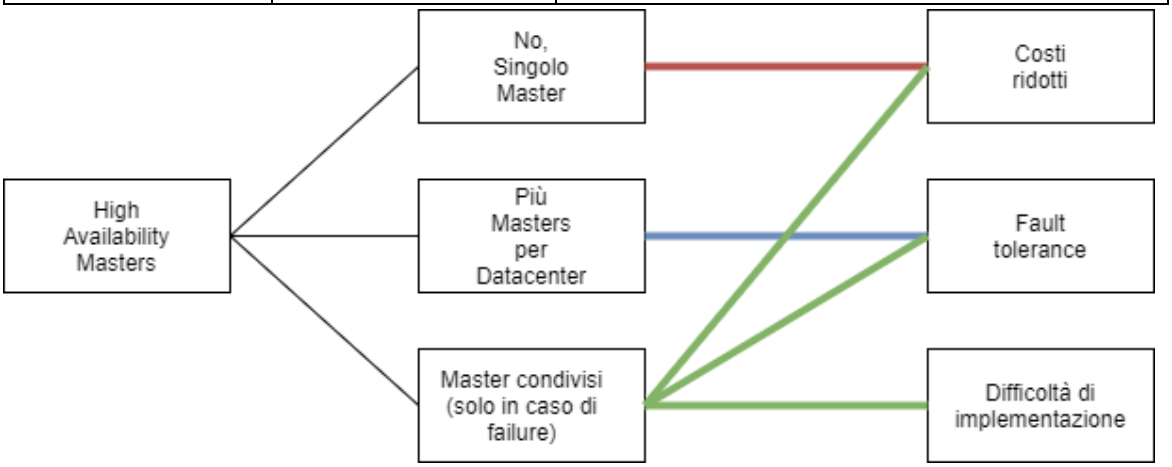
High availability master nodes

Per implementare dei master node più fault tolerant avremmo bisogno di un altro servizio delegato ai master node stessi: Pacemaker il quale permette di eseguire
un deployment ridondante dei master node ed inoltre di gestire un Virtual IP univoco per tutti i master node (Single point of entry, ma non Single point of failure).
Per mantenere bassi i costi si è deciso di mantenere un solo master node per data center che, in caso di crash, sarà sostituito momentaneamente dal master node dell'altro data center della factory, in caso di fallimenti multipli si potrebbe fare un deployment rapido di un altro master al posto di un worker, sacrificando un po' delle prestazioni dei servizi T2 che girano in locale.
Di seguito un semplice diagramma ad alto livello dell'architettura.

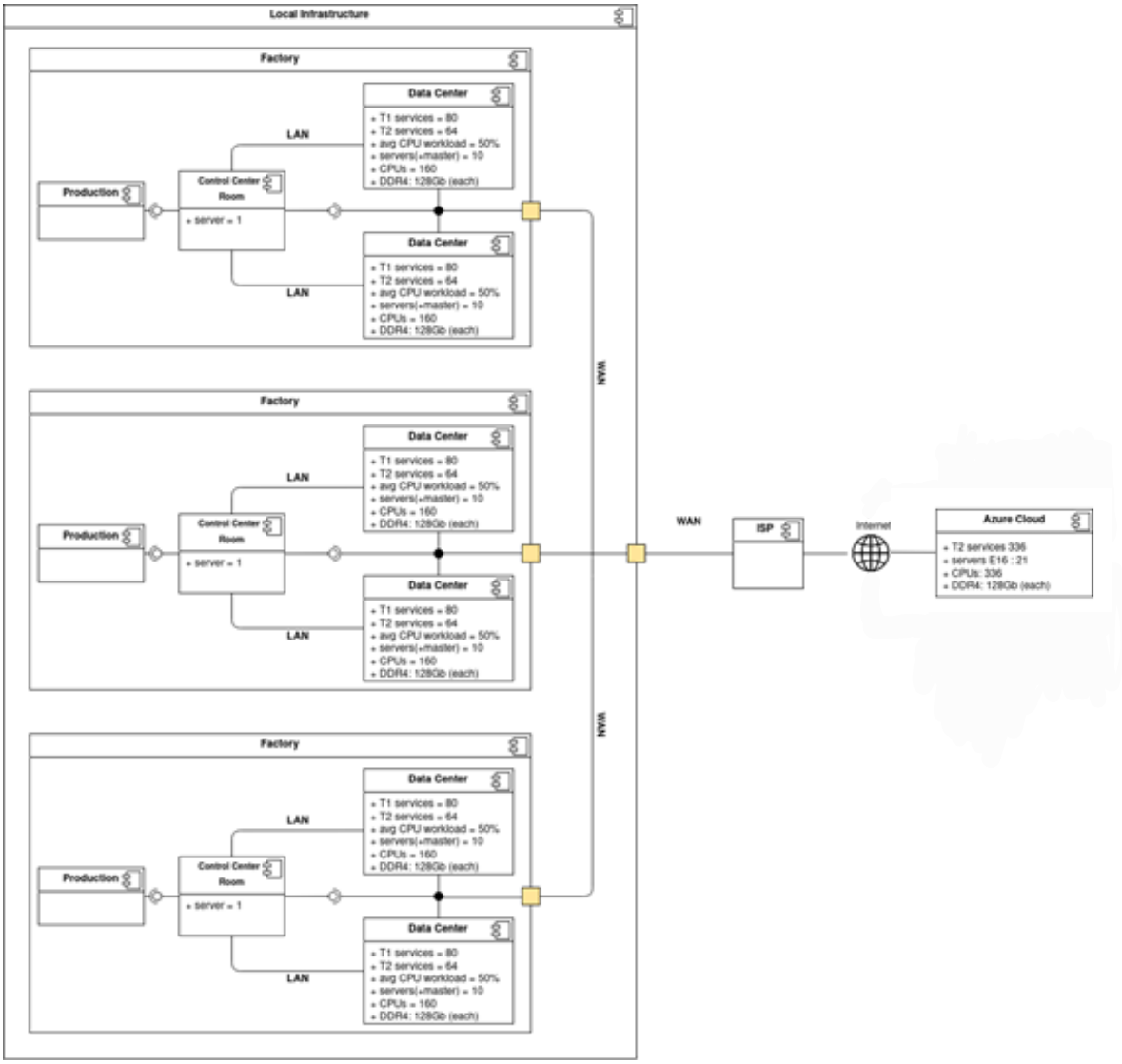


Concern		Implementazione del master node
Alternative(s)		1. Un solo master per datacenter 2. Master ridondanti in ogni datacenter 3. I master dei due datacenter sono intercambiabili
Ranking criteria		4. Costi relativi all’acquisto dell’hardware 5. Disponibilità. I master devono avere disponibilità pari ai servizi T1
Architectural decision	**Identifier**	14
	Description	I master dei due datacenter sono intercambiabili, nel caso in cui uno dei master dovesse cadere l’altro

		prenderebbe il suo posto, nel caso di fallimento catastrofico di entrambi i master, un nuovo master avverrà il deploy da parte di pacemaker al posto di uno dei server a discapito delle prestazioni dei servizi T2
	Rationale	Senza il master node nessuno dei servizi di kubernetes (OpenShift) potrà operare motivo per il quale si è deciso per la scelta più complessa ma che massimizza la fault tolerance



Architecture diagram

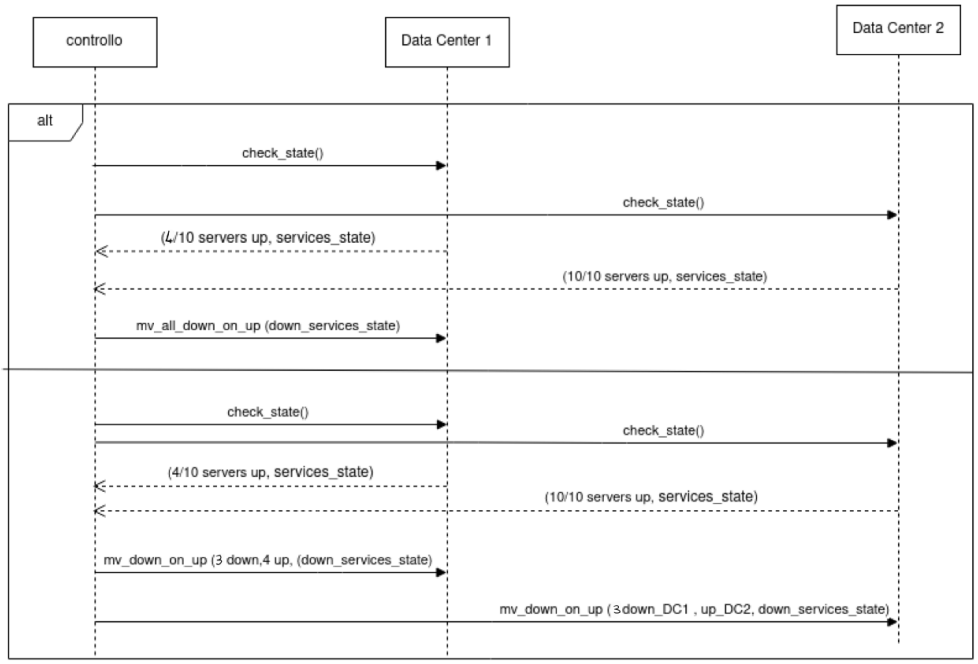


Il Diagramma rappresenta l'intera architettura del sistema, composta di **3 factory**, connesse tramite un servizio WAN, e ogni factory presenta **2 Data Center**, una **control center room** e un area di **Production**, connessi tramite LAN. È inoltre presente un servizio **Cloud di Azure** esterno all' Infrastruttura.

Per ogni data center è indicato: il **numero di server**, il **numero di cpu**, la loro **memoria DDR4**, il **numero di servizi** che gestisce, la loro **tipologia** e **l'utilizzo medio di cpu** del datacenter.

L'interfaccia fornita ai Data Center dalla Control center room permette la sua comunicazione sia con i data center nella sua factory che con le control center room di altre factory.

Crash di server in un Datacenter



Un data center può continuare a gestire i servizi che gli sono stati assegnati fino al crash della **metà dei suoi server meno uno**, a causa del **master node server,** poichè, ognuno lavora al **50% di workload medio**, perciò se 4 diventano non disponibili, i rimanenti si possono occupare dei servizi crashati arrivando al **100% di CPU workload medio**. Questa situazione è descritta nella **prima alternativa** del sequence diagram dove 4 server su 10 diventano non disponibili per il data center 1.

È da notare che la funzione **mv** *all* **down_up()** ha come parametro lo **stato dei servizi** crashati nel momento in cui hanno smesso di funzionare, in modo che i server che dovranno riprenderne l'esecuzione potranno farlo **dal punto in cui sono stati interrotti**. Lo stato dei servizi viene mandato in risposta dal data center al **check_state()** effettuato dal controllo.

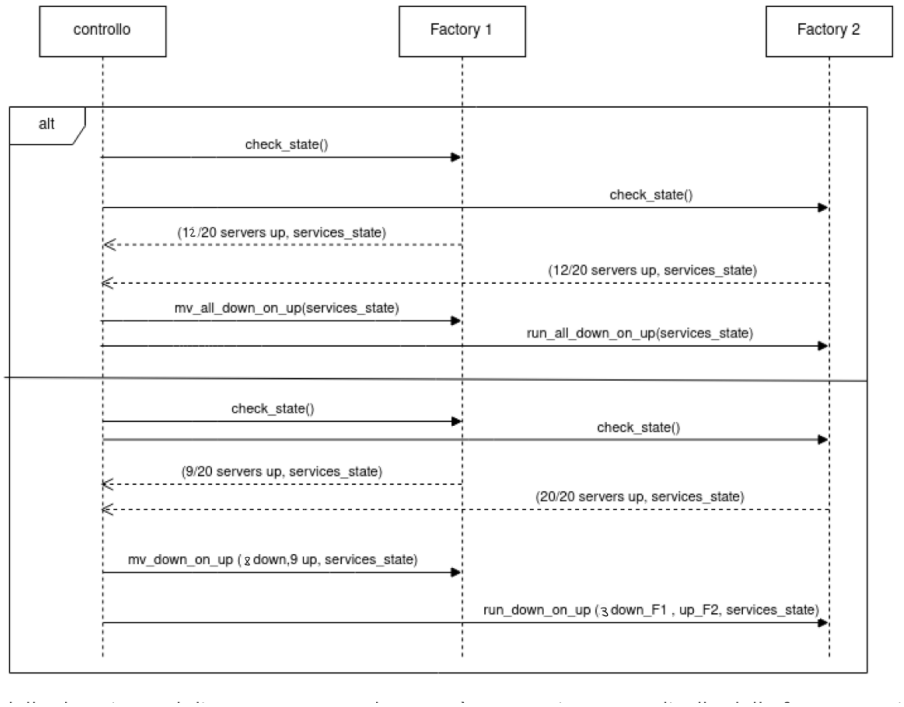
Nel caso di un failure della **metà o più dei server**, i servizi interrotti vengono assegnati nel seguente modo:

ai **server rimasti** dello stesso datacenter vengono assegnati **tutti quelli che è in grado di gestire**, ovvero ricordando che lavorano ognuno al 50%, segue che ogni server può gestire il lavoro di due server, tranne per il master node server (ad

esempio se crashano 6 server, ne restano 4 up, che possono ospitare il lavoro di altri 3 prima di raggiungere il 100% di CPU workload medio)

ai server dell'**altro data center nella stessa factory** vengono assegnati i servizi rimanenti (ad esempio, nel caso di crash di 6 server nel data center 1, questo può continuare a gestirne 3, e i 3 rimanenti vengono assegnati al data center 2)

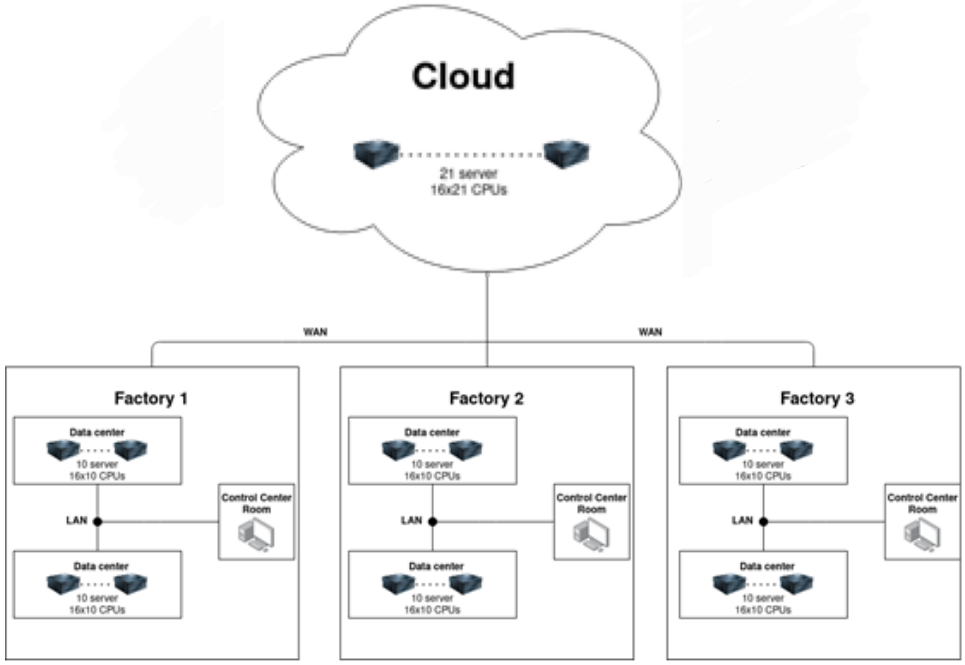
Crash di server in una Factory



Il modello descritto nel diagramma precedente può essere riportato a livello delle factory: se in una factory crashano **meno della metà meno due** (un master node per ogni data center) dei server ospitati, questa **può ospitare tale parte crashata nella parte rimanente** dei server, arrivando al **100% di workload medio**; se invece ne crashano altri, può assegnare i servizi che non riesce a gestire a un' **altra factory** nello stesso modo in cui un data center li sposta ad un altro nella seconda alternativa del **sequence precedente**.

È da specificare, che i servizi che vengono spostati in un altra factory, sono sempre **divisi equamente** tra le altre due factory. in questo se è una factory intera a crashare, le altre due factory arriveranno a un cpu workload del **75% ognuna, anzichè una al 50% e una al 100%.**

Infrastructure view



Hardware locale

- 3 Factory
- 2 Data Center ogni factory
- 9 server ogni Data Center
- 1 master node server ogni Data center
- 16 CPU per ogni server
- 160 servizi T1 per ogni factory
- 720 servizi T2 totali

Ogni data center ha la capacità di 144 CPU, e deve gestire 80 servizi T1 che lavorano con il 40% di workload, perciò il workload medio, assegnando una CPU a ogni servizio, è pari a:

$$80*40+80*0144=22\%$$

dove 80 sono il numero di servizi assegnati e 40 è la percentuale di workload utilizzato da ognuno. Segue che rispettando il requisito che impone un workload medio del 50%, possiamo ulizzare il **28% residuo per osipitare dei servizi T2** il cui CPU workload è del 60%, il numero massimo di servizi T2 gestibili da un data center è dato da:

$$40*80+60*n+144-n144=50=>n=64$$

di conseguenza, ogni data center, assegnando un servizio a CPU, avrà assegnati **80 servizi T1**, **64 servizi T2**, mantenendo un workload medio di:

$$40*80+60*64144=49\%$$

inoltre, cosi facendo nel caso del fault di un intero data center, il suo 50% di lavoro sarà trasferito sull'altro data center, il quale arriverà ad avere il **100% di CPU workload medio**.

Un altro vantaggio di questa scelta riguarda il **fault di un intera Factoy**, infatti il carico di lavoro dei due data center, può essere spartito tra i data center delle altre factory.

Vengono quindi assegnati a **ogni data center 64 servizi T2**, perciò dei **720 previsti** dalla specifica, siamo in grado di gestirne localmente **64*6=384**. Riduciamo così la spesa per il servizio cloud, scelto per gestire i **336 servizi T2 rimanenti.**

Inoltre sono presenti 3 server di controllo, uno per ogni factory.

Cloud

- 336 servizi T2
- 16 CPU per ogni server

Dato che ogni servizio T2 utilizza il 60% di workload, abbiamo deciso di assegnare un servizio per ogni CPU, di conseguenza abbiamo bisogno di 336 CPU, perciò dato che ogni server ne ha 16, abbiamo bisogno di **336/16 = 21 server**.[Ritorno a capo del testo]Abbiamo scelto di utilizzare i **server E16 di Azure Cloud.**

Avendo 3 factory che lavorano al 50%, è possibile gestire un crash globale di una 20 server, ovvero il crash di metà della struttura, poiché la metà rimanente è in grado di gestire tutto il carico della struttura.

Nel caso crashino più di metà server, iniziamo a degradare le prestazioni dei servizi T2 fino a poter gestire tutti i servizi assegnati localmente con un singola factory. Perciò nel caso crashino due factory abbiamo 480 T1 e 372 T2 da assegnare alla factory rimanente, la quale ha 288 cpu a disposizione (18 server, tolti i due master node), assegnamo i servizi nel seguente modo:

- 2 T1 a 12 CPU che lavoreranno all’ 80% di workload;
- 2 T1 e 1 T2 al 20% di workload a 180 CPU che lavoreranno al 100% di cpu workload;
- 1 T1 e 2 T2 al 30% di workload a 96 CPU che lavoreranno al 100% di cpu workload.

In questo modo la factory avrà un cpu workload medio del 99% dato da:

(12*80+180*100+96*100)/288 = 99%

Il **limite di CPU richieste a gestire tutti i servizi T1(al 40%) e T2 (al 20%) localmente è 267**, che corrispondono a 18 (+2 master), esattamente 2 data center, cioè una factory, e la distribuzione migliore dei processi è proprio quella descritta precedentemente.

Nel caso peggiore in cui ci sia un crash dei server dell'ultima factory rimasta, il massimo che possiamo fare è mantenere localmente solo i servizi T1, e il minimo numero di cpu che servono a gestirli è 240 (2 servizi T1 a cpu) che corrispondono a 15 server e 2 server master node, poiché il numero di server richiesti è necessariamente spartito tra due data center, di conseguenza dobbiamo spostare tutti i servizi T2 (384) sul cloud, dove sono già presenti 336 servizi T2. Assegniamo i 720 servizi T2 nel seguente modo:

- 288 CPU gestiranno 576 servizi T2, due servizi per ogni CPU che lavorano al 50% di workload ognuno, raggiungendo il 100% di workload per ogni CPU
- Le rimanenti 48 CPU gestiranno 144 servizi T2, tre ogni Cpu che lavorano al 33% di workload ognuno, raggiungendo il 100% di workload per ogni CPU

Costi

L'hardware locale è composto di **60 server**, di 16 CPU ognuno e 128GB di memoria DDR4. Il costo di un server è di **12000 USD**, il mantenimento invece ammonta a **1200 USD l'anno**, ed è previsto un **rimpiazzo ogni 3 anni.**

Segue che la spesa per l'hardware locale ammonta a:

12000*60 + 1200*60*3 = 936000 USD

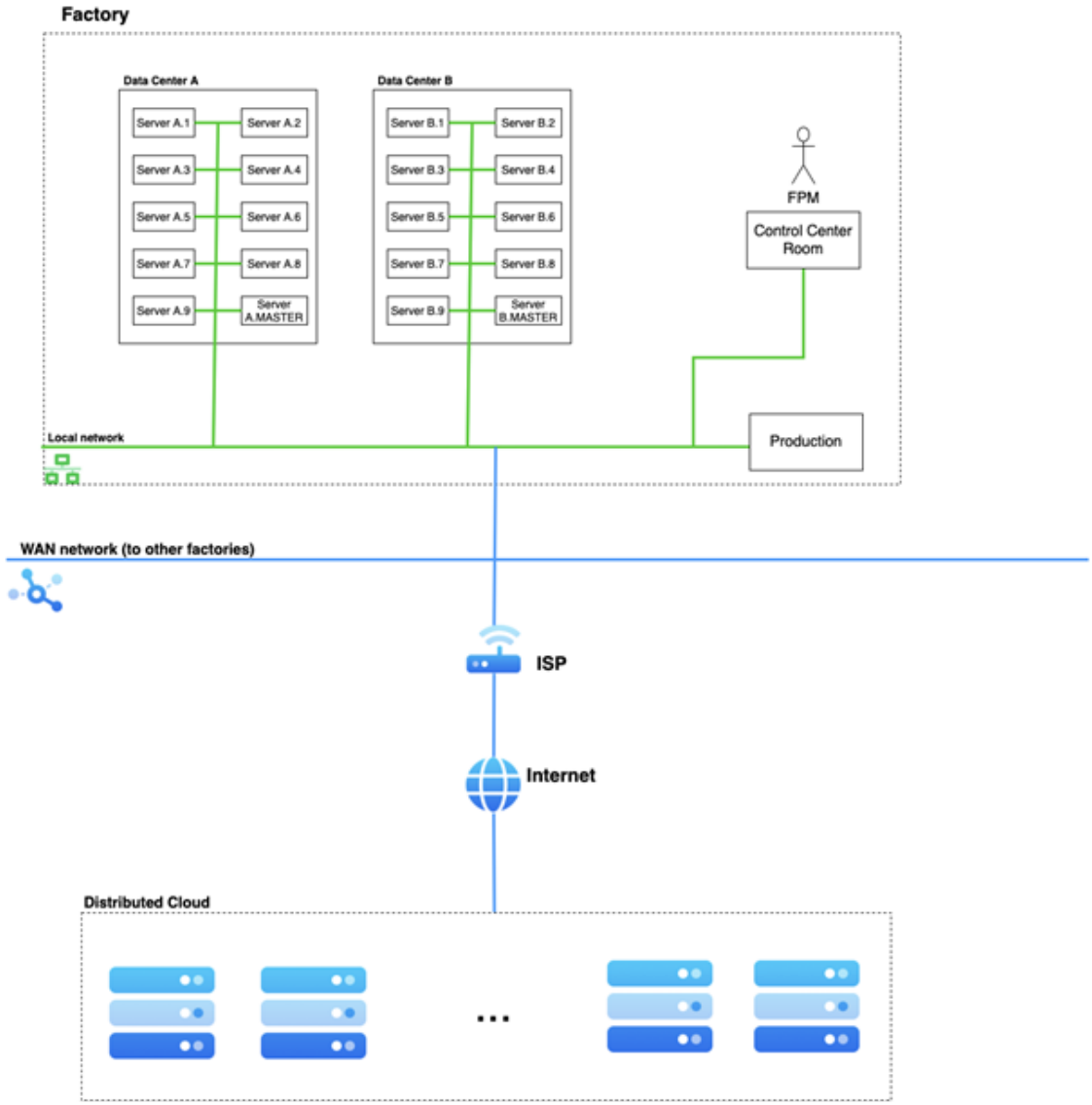
Il servizio cloud invece, composto da 21 server E16 di Azure, con 16 CPU ognuno e 128GB di memoria DDR4, sono stimati 847.5 USD al mese, per un periodo di 3 anni. [Ritorno a capo del testo]Il costo totale ammonta a;

21*848=17808 USD

Perciò la spesa totale, ogni 3 anni ammonta a:

936000+17808=953808 USD

Factory Production Manager Viewpoint



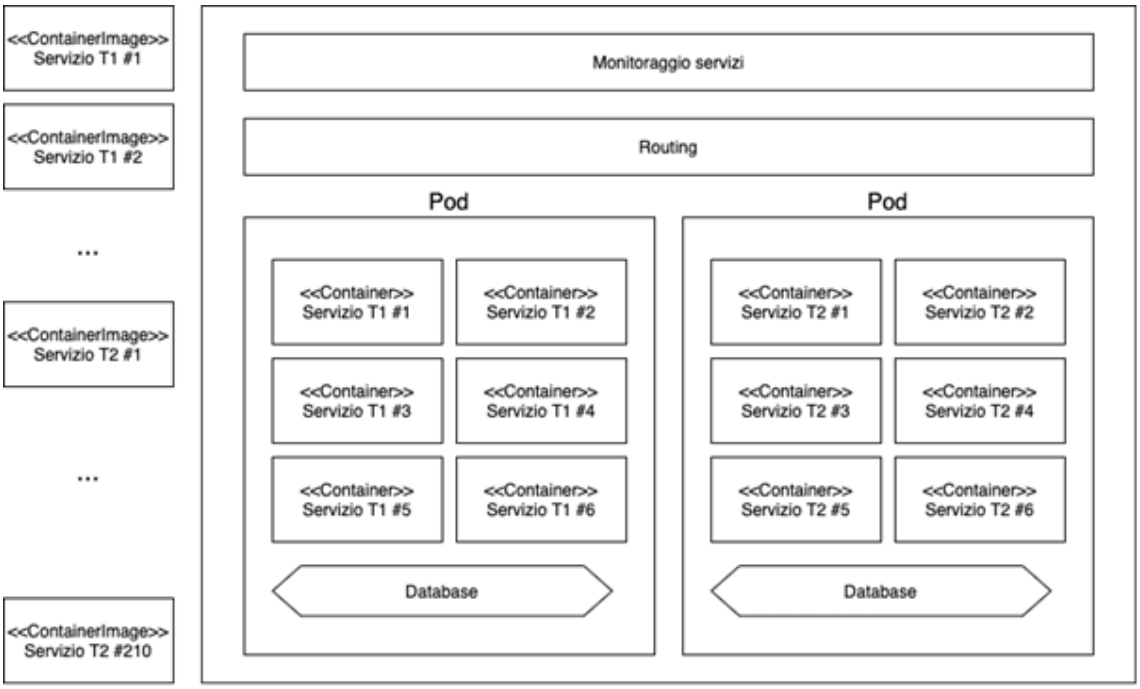
All’interno di questo schema è possibile vedere l’architettura ad alto livello della singola fabbrica e i suoi collegamenti con la rete esterna.
Questo schema rappresenta il punto di vista del Factory Production Manager, fornendogli le informazioni necessarie atte a capire la struttura generale della fabbrica, e l’ubicazione dei servizi di interesse.
I servizi di tipo T1 vengono eseguiti all’interno della fabbrica.
Alla rete locale vengono collegati tutti i server, oltre alla Production, cioè l’insieme delle macchine che si occupano di produrre i chip, e la Control Center Room, una stanza dedicata al controllo dell’andamento della produzione.

La fabbrica è connessa alle altre fabbriche tramite una rete WAN privata, che a sua volta è connessa ad Internet mediante l’ISP.
È inoltre presente un cloud di terze parti dove vengono eseguiti i servizi T2, monitorabili anch’essi dalla Control Center Room.

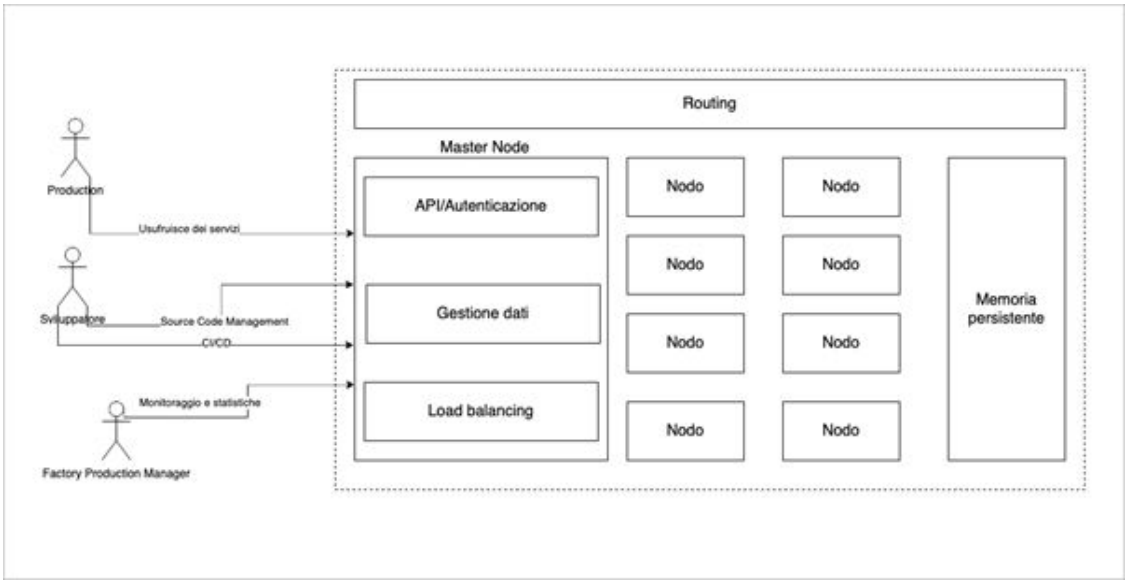
Descrizione di come gli ASR sono conformi all’architettura proposta

Disponibilità al 99,999%: Viene raggiunta tramite la ridondanza dei data center, poiché ogni server di un data center ha un 50% di utilizzo di CPU medio, e nel caso in cui un data center smetta di funzionare, il carico di lavoro viene momentaneamente trasferito all’altro data center.
Sicurezza dei dati: Tutti i dati dei servizi T1 non vengono condivisi al di fuori della fabbrica, e sono accessibili solamente dalla Control Center Room. Per quanto riguarda il trasferimento dei dati tra due fabbriche, questa operazione avviene tramite la rete WAN privata.
Performance: a livello hardware, i server vengono aggiornati ogni 3 anni, dunque possiamo aspettarci delle performance all’avanguardia con l’attuale stato dell’arte per quanto riguarda le prestazioni hardware; a livello software, kubernetes ed open shift garantiscono di sfruttare al meglio l’hardware

IT Software Development Manager viewpoint



In questo schema è possibile vedere l’architettura di un generico server di un data center. All’interno di ogni server è presente un software che si occupa del monitoraggio dei servizi, cioè la possibilità di controllare lo stato dei servizi in esecuzione sulla macchina, e del routing, che si occupa di reindirizzare le richieste nel pod corretto; inoltre, sono presenti i vari pod per eseguire i servizi. All’interno di ogni pod i servizi devono appartenere tutti alla stessa categoria.



Questo schema mostra l’architettura software di un data center. Ogni data center ha un master node, che si occupa dell’autenticazione degli utenti che lavorano al sistema, un servizio per la gestione dei dati che saranno memorizzati nella memoria persistente, e un servizio per il load balancing. I nodi di un data center si occupano di eseguire i servizi T1 e T2, mentre le chiamate vengono gestite tramite il layer di routing.
Gli attori che abbiamo individuato sono: lo sviluppatore, poiché deve effettuare operazioni di Source Code Management e Continuous Integration e Continuous Deployment. L’attore “Factory Production Manager”, tramite la Control Center Room, può monitorare l’andamento del data center, connettendosi al master node. Infine, il reparto Production si occupa effettivamente di usufruire dei servizi forniti da un data center.

Conclusioni

	# Server locali	prezzo (cad.)	Utilizzo medio CPU locale	# Server su Cloud	Utilizzo medio CPU su cloud	Prezzo (cad. al mese)	max # server su Cloud pay as you go	prezzo server cloud pay as you go (cad. al mese)	prezzo Totale
480 T1+ 384 T2 locali, 336 T2 Cloud	60	12000 USD	49%	21	dal 60% al 100%	857			1583892 USD
Tutti i servizi in locale	60	12000 USD	72%			857			936000 USD
T1 in locale e T2 su cloud	60	12000 USD	22%	45	60%	857			2108340 USD
480 T1+ 384 T2 locali, 336 T2 Cloud + pay as you go	60	12000 USD	49%	21	60%	857	24	1354	2753748 USD

Nella tabella sono riassunte le alternative di design, ed è possibile effettuare un rapido confronto tra di esse.
Si può notare che che nell’alternativa più economica, in cui tutti i servizi sono allocati localmente, il workload medio di CPU supera il 50% di conseguenza sarebbe impossibile raggiungere la fault tolerance richiesta.
Invece l’alternativa adottata risulta essere la più economica tra quelle conformi ai requisiti, inoltre la nostra scelta è quella che raggiunge la massima fault tolerance tra le varie alternative.