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

Introduction

Challenges/Risk Analysis 3

Spec Refinement 5

Design Decisions 7

- Esecuzione dei servizi di tipo T2 7
- Cloud pubblico 8
- Fault Tolerance 10
- Bilanciamento del workload 12
- Master Node 14
  - High availability master nodes 14
- Architecture diagram 17
- Crash di server in un Datacenter 18
- Crash di server in una Factory 19

Infrastructure view 20

- Hardware locale 20
- Cloud 21
- Costi 22

Factory Production Manager Viewpoint 24

Description of how ASR are met by proposed architecture 25

IT Software Development Manager viewpoint 26

Introduzione

	# 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
480 T1+ 480 T2 locali, 336 T2 Cloud	30	12000 USD	100%	8	100%	857	16	1354	805824 USD

Nella tabella sono riassunte le alternative di design, ed è possibile effettuare un rapido confronto tra di esse.
Si può notare che l’alternativa scelta (quella evidenziata), risulta essere anche la più economica.
Nonostante l’utilizzo dell’hardware locale sia del 100% è possibile garantire una notevole fault tolerance in caso di crash, tramite lo spostamento momentaneo dei servizi T2 coinvolti su un cloud pay as you go e la suddivisione dei servizi T1 falliti tra i server operativi rimanenti.
In questo modo è possibile continuare a gestire la totalità dei servizi T1 anche nel caso del fallimento di metà dell’hardware locale, senza impattare notevolmente sui costi rispetto un'alternativa che utilizzi più server locali.

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**; è risultato che 15 server sono il minimo numero per gestirli

Gestione Fault Tolerance	9/12/2020	10/12/2020	**La fault tolerance** è gestita principalmente a livello hardware, utilizzando ogni cpu al **100%, e prevedendo che ogni cpu abbia assegnato un servizio T1 e un servizio T2;** un server può **gestire l’eventuale fallimento** di un altro nel seguente modo: • spostando sia i **T2 del server S1 locale fallito** sia i **T2 del server S2 locale di rimpiazzo** su un server **S3 di un cloud pay as you go**, degradandone le prestazioni al **50%** • spostando sul server **S2 di rimpiazzo** i T1 del server **S1 fallito** In questo modo bastano 15 server Worker locali (non master node) operativi per gestire la totalità dei T1 e 16 server su cloud pay as you go per gestire tutti i T2, precedentemente locali, al 50% di workload
Bilanciamento Workload CPU	9/12/2020	9/01/2021	L’**utilizzo medio delle cpu di ogni server è al 100%** abbiamo così ottimizzato il **numero di servizi T1 e T2 gestibili dall’hardware locale. e minimizzato i servizi T2 da gestire su cloud. ogni factory** dovrà gestire **160 T1** e, avendo a disposizione **160 CPU** operative per factory, possiamo allocare anche **160 servizi T2,** Inoltre, in caso di crash il workload di tutti servizi T2 locali può essere degradato al 50%, per gestirli tutti sul cloudpay as you go, e poter gestire in 15 server locali (esclusi due master node) tutti i servizi T1.

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. I servizi **T1** lavorano con il **40%** di cpu workload totale
    b. 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 **6 server di cui un master node server**
    a. I server possono avere più di una cpu

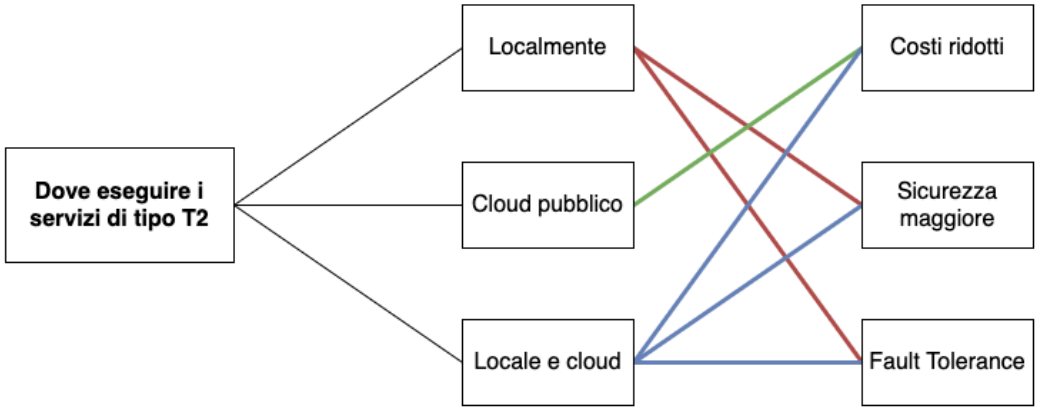
15. Nel caso di **crash** di un data center, deve essere garantita abbastanza potenza di calcolo per continuare a gestire tutti quanti i servizi richiesti

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

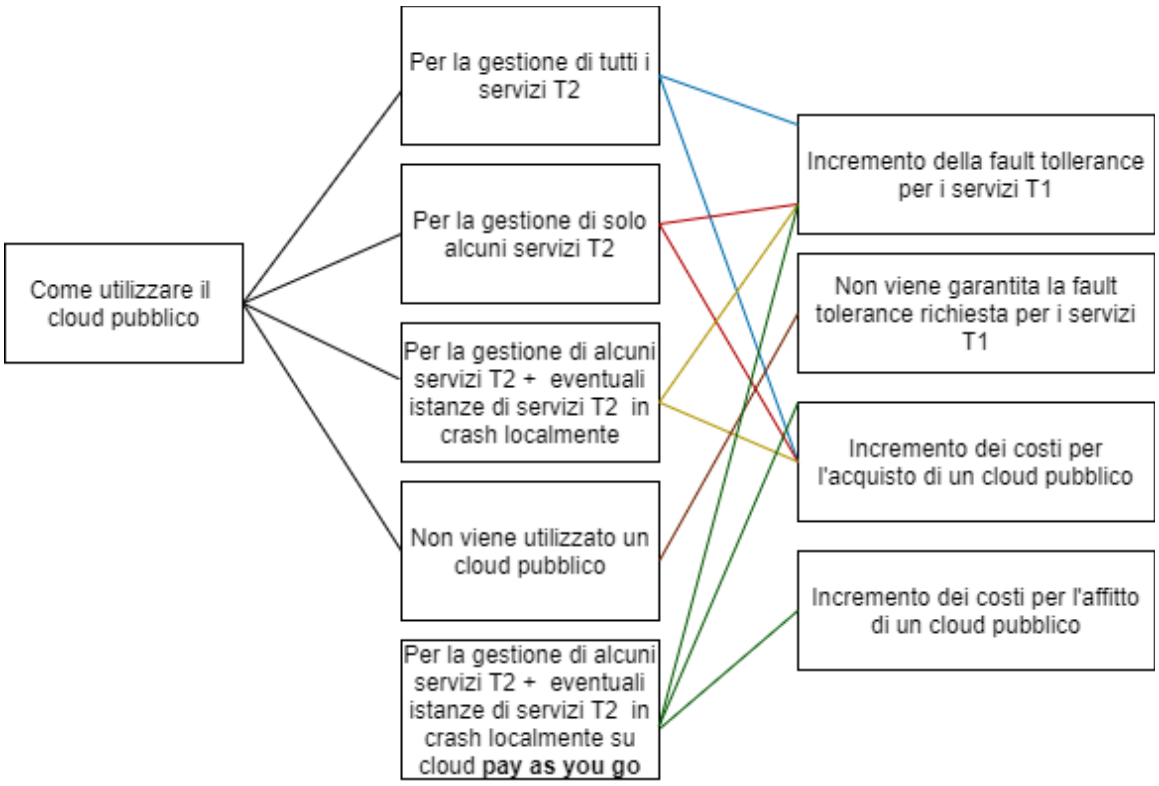


Cloud pubblico

Utilizzo di un cloud pubblico per la gestione di parte dei servizi T2, ed utilizzo di uno pay as you go 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	**Identifier**	6b

decision	**Description**	Utilizzo di un cloud pubblico per la gestione di alcuni dei servizi T2, ed utilizzo di uno pay as you go per 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 e uno pay as you go per garantire un sufficiente hardware interno per la gestione dei servizi T1, anche in caso di grandi fallimenti.

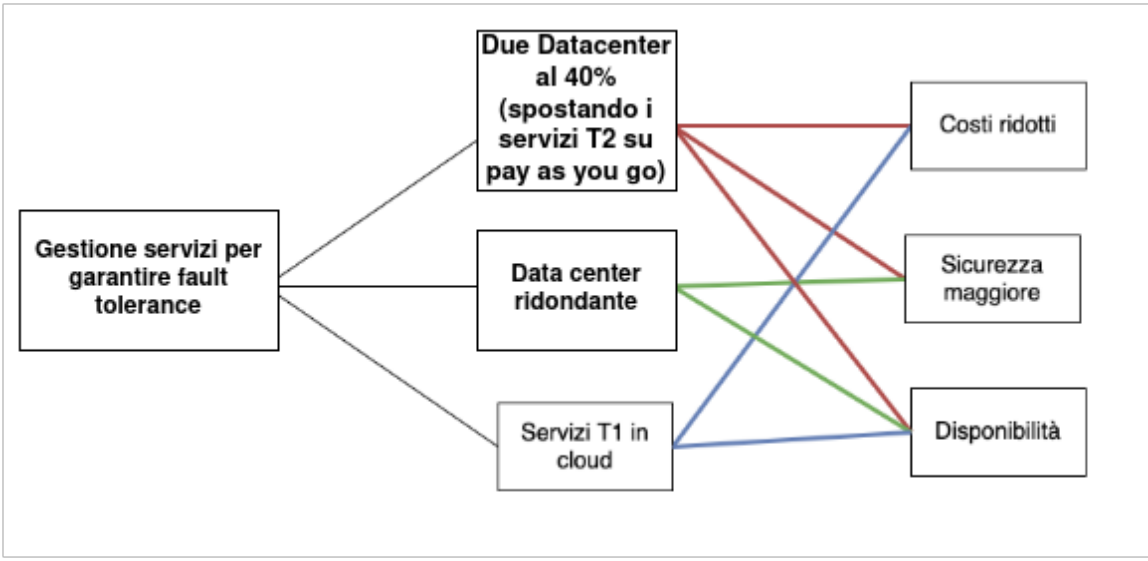


Fault Tolerance

La Fault tolerance viene garantita principalmente a livello hardware, prevedendo che ogni factory abbia 2 data center.
Nel caso in cui un data center smetta di essere operativo, l’altro può eseguire i servizi di entrambi i data center spostando i servizi T2, poichè è possibile degradare le prestazioni dei servizi T2 locali al 50% per spostarli tutti sul cloud pay as you god, e poter ospitare tutti i servi T1 localmente in soli 15 server, in caso di crash.

Concern		Garantire operabilità dei servizi T1 al 99.999%
Alternative(s)		1. Due datacenter con al 100% del carico, con spostamento dei servizi T2 su cloud a prestazioni ridotte 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 al 100% di carico e in caso di crash sposteranno i loro servizi T2 su cloud pay as you go , imponendo il loro workload al 50%, in modo tale da garantire, nel caso in cui un datacenter fallisca, che l’altro possa prendersi in carico il suo workload

	Rationale	Due data center con 100% di workload, con spostamento dei servizi T2 su cloud pay as you in caso di crash



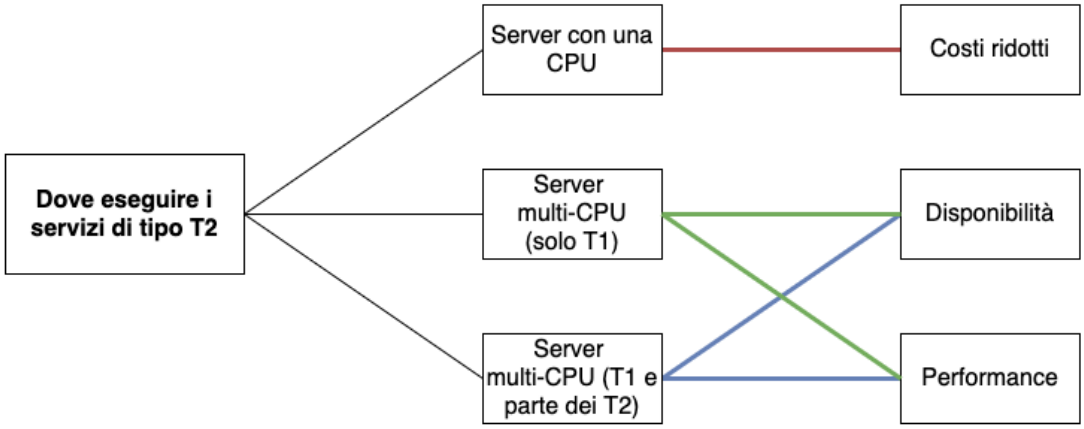
Bilanciamento del workload

Ogni server della factory è multi-cpu, in modo tale che ogni factory abbia il 100% di workload medio, suddiviso per 160 servizi T1 e 160 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)
- un servizio T1 e uno T2 (40% + 60% di cpu workload)
- Due servizi T2 al 50% di workload ognuno.

Inoltre le prestazioni dei T2 possono essere degradate al 50% di workload, così da poter spostare in una cpu del cloud pay as you go 2 servizi T2, lasciando spazio a una cpu locale per gestire due servizi T1 all'80% di workload.

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:

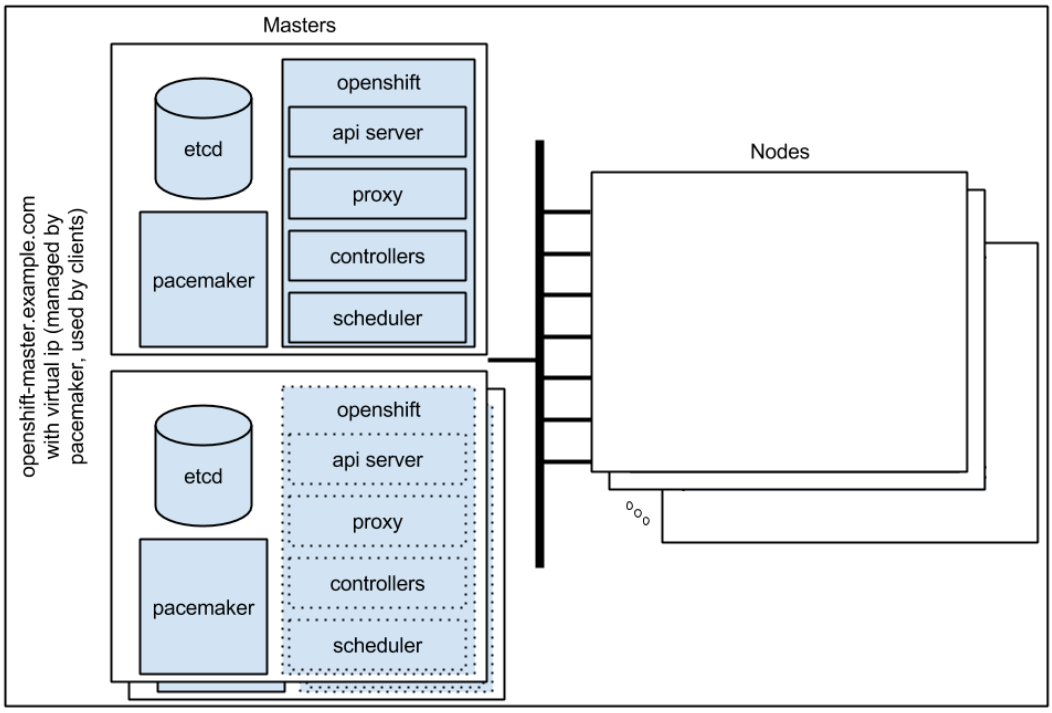
- La gestione delle API di kubernetes (configurazione dei pods, assegnazione dei pods ai nodi, sincronizzazione tra pods), etcd (store permanente di dati chiave-valore che mantiene lo stato del master in caso esso debba essere spostato o riavviato),
- controller manager (controlla i cambiamenti in etcd 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.
Il Pacemaker 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 parte delle prestazioni di servizi T2 che verranno spostati sul cloud e trasferendo i suoi servizi T1 ad un altro worker

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

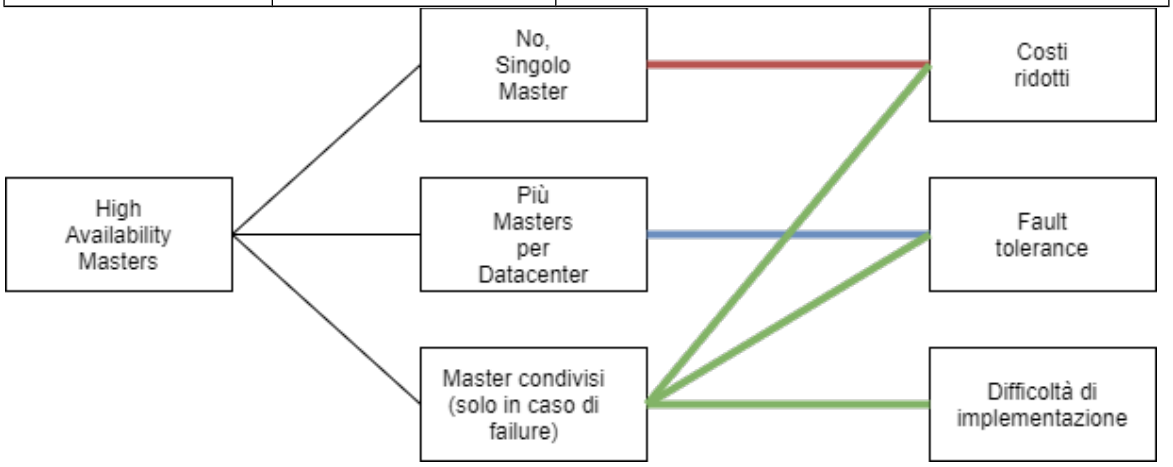
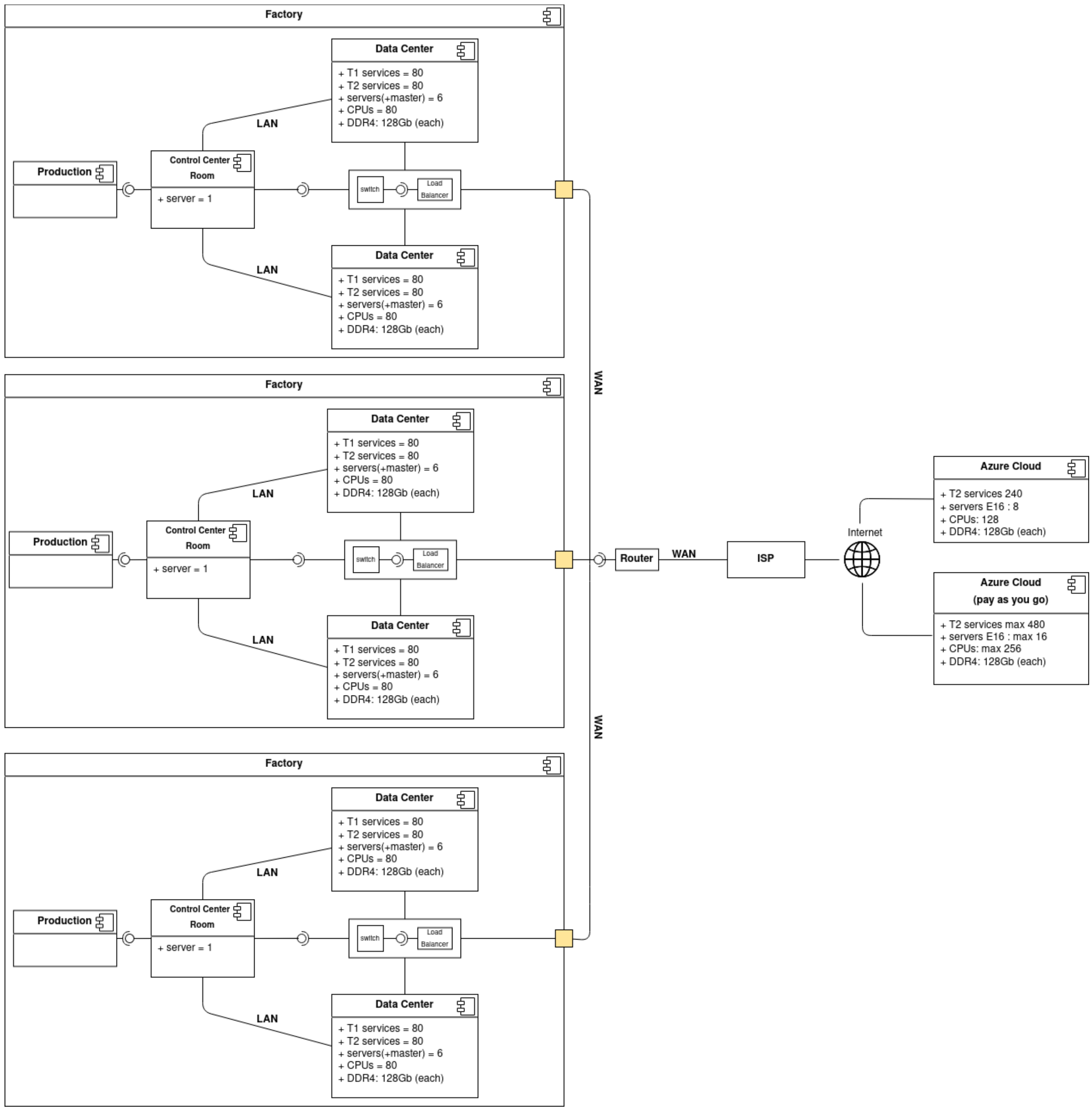


Diagramma Architetturale

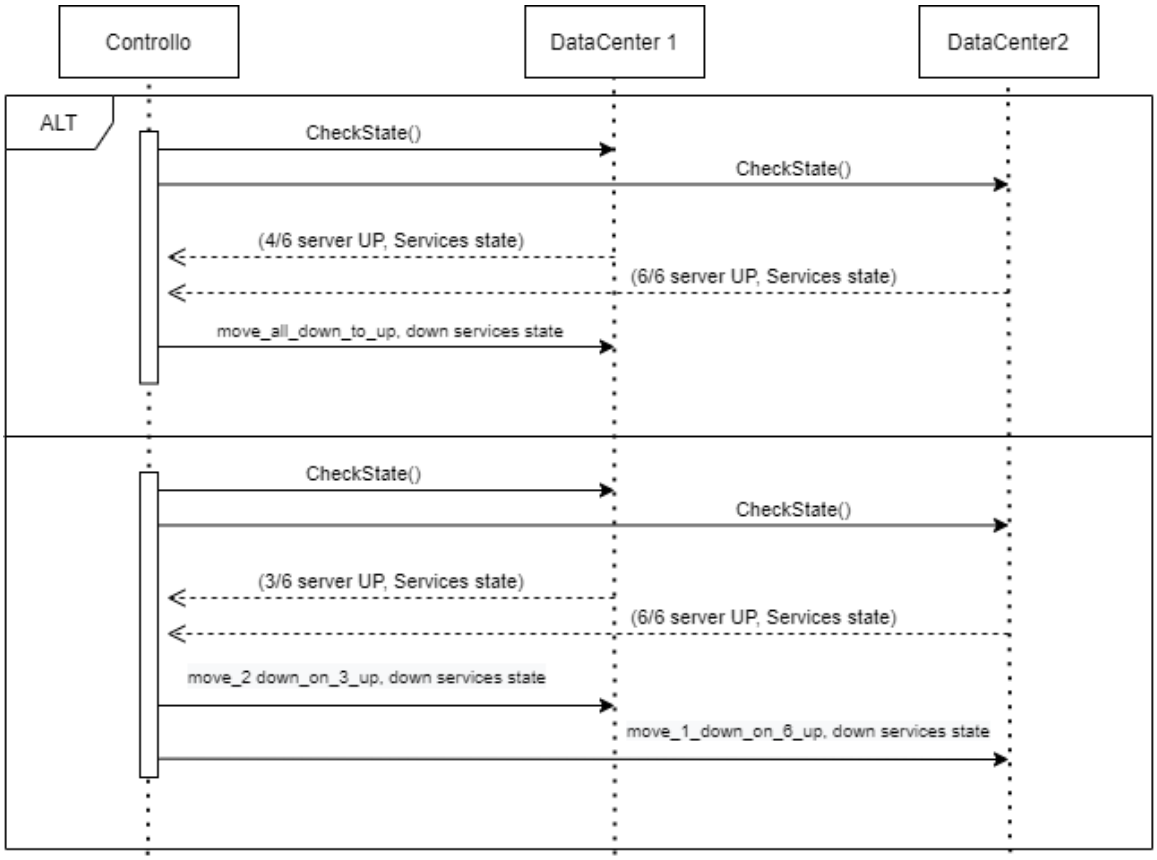


Il Diagramma rappresenta l'intera architettura del sistema, composta di **3 factory**, connesse tramite un servizio WAN, e ogni factory presenta **2 Data Center**, una **control center room** e un area di **Production**, connessi tramite LAN. È inoltre presente un servizio **Cloud di Azure** esterno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

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,** poiché, ogni **server worker** lavora al **100% di workload medio**, perciò se 2 su 5 diventano non disponibili, i rimanenti 3 possono occuparsi dei servizi crashati nel seguente modo:

- spostando tutti i servizi T2 coinvolti su cloud (sia dei server crashati sia su quelli di rimpiazzo) degradando il loro utilizzo di CPU al 50%
- spostando poi tutti i servizi T1 dei due server falliti su due dei tre server operativi, arrivando ad un **CPU workload dell’86%** (due server all’80%

poiché con due servizi T1, e uno al 100% poiché ancora con un servizio T1 e un T2).

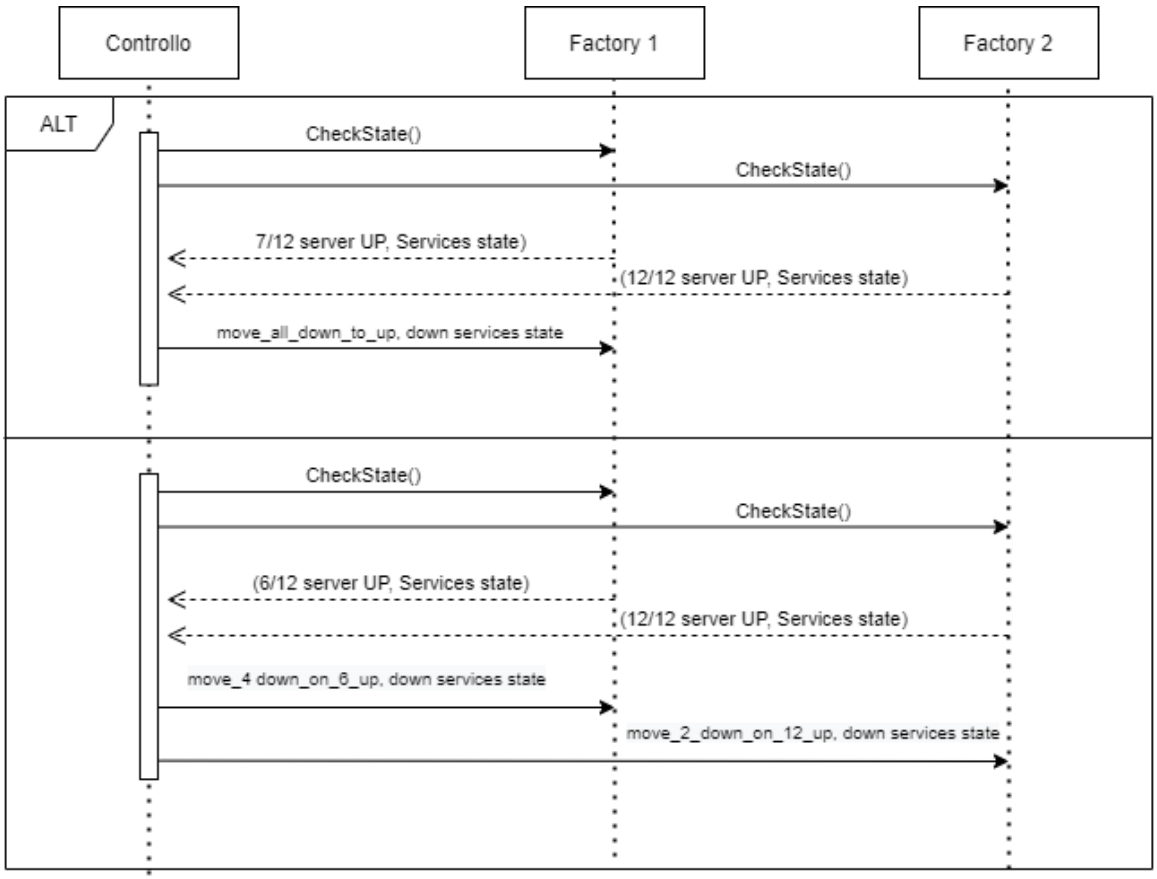
Questa situazione è descritta nella **prima alternativa** del sequence diagram dove 2 server su 6 diventano non disponibili per il data center 1.

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 del fallimento della **metà o più server** di un datacenter, i servizi interrotti vengono assegnati nel seguente modo:

- i servizi T2 coinvolti vengono spostati su cloud pay as you go come descritto precedentemente;
- ai **server rimasti** dello stesso datacenter vengono assegnati **tutti quelli che è in grado di gestire**, ovvero ricordando che, tolti i servizi T2, lavorano ognuno al 40% segue che ogni server può gestire il lavoro di due, tranne ovviamente per il master node.
- ai server dell'**altro data center nella stessa factory** vengono assegnati i servizi rimanenti (ad esempio, nel caso di crash di 3 server nel data center 1, questo può continuare a gestirne 2, e il rimanente verrà assegnato al data center 2)

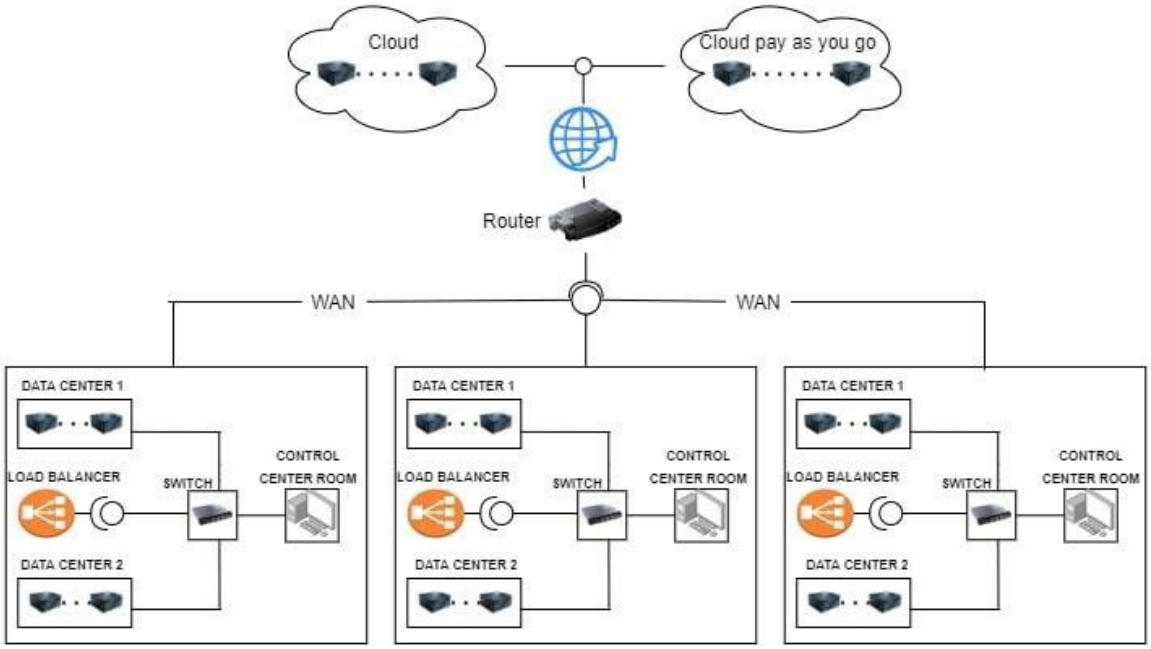
Crash di server in una Factory



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 worker ospitati, questa **può ospitare la parte crashata nella parte rimanente** dei server spostando i servizi T2 coinvolti sul cloud pay as you go degradandone al 50% le prestazioni**.**
Se 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 diagram precedente**.

È da specificare, che i servizi che vengono spostati in un altra factory, sono sempre **divisi equamente** tra le altre, al fine di avere un workload distribuito.

Infrastructure view



Hardware

- Tutti i T1 in locale al 40% di workload
- 480 servizi T2 in locale al 60% di workload
- 240 servizi T2 sul cloud al 50% di workload
- Cpu workload medio locale al di sotto del 100%

Mettendo un T1 e un T2 a Cpu su server da 16 cpu, abbiamo bisogno di 480/16 = 30 server ognuno al 100% di workload.

dato che abbiamo 2 data center a factory e ognuno ha un server master node possiamo assegnare 6 server a data center per un totale di 36 server spartiti tra le 3 factory.

Imponendo i servizi T2 al 50% di workload possiamo metterne due su ogni CPU e utilizzando server E16 di Azure da 16 CPU, bastano (240/2)/16=8 server al 100% di workload.

In caso del crash locale possiamo mettere al più due servizi T1 per cpu e spostare i 480 T2 su cloud, di conseguenza basta metà dell'hardware locale per gestire tutti i T1, all'80% di workload totale, mentre abbassando il CPU workload dei T2 rimossi al 50% possiamo utilizzare 16 server E16 di Azure pay as you go, assegnando a ognuno 2 servizi T2.

Costi

L'hardware locale è composto di **36 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:

$$12000*36 + 1200*36*3 = 561600 \ USD$$

Il servizio cloud invece, composto da 8 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.
Il costo totale ammonta a:

$$8*848*12*3 = 244224 \ USD$$

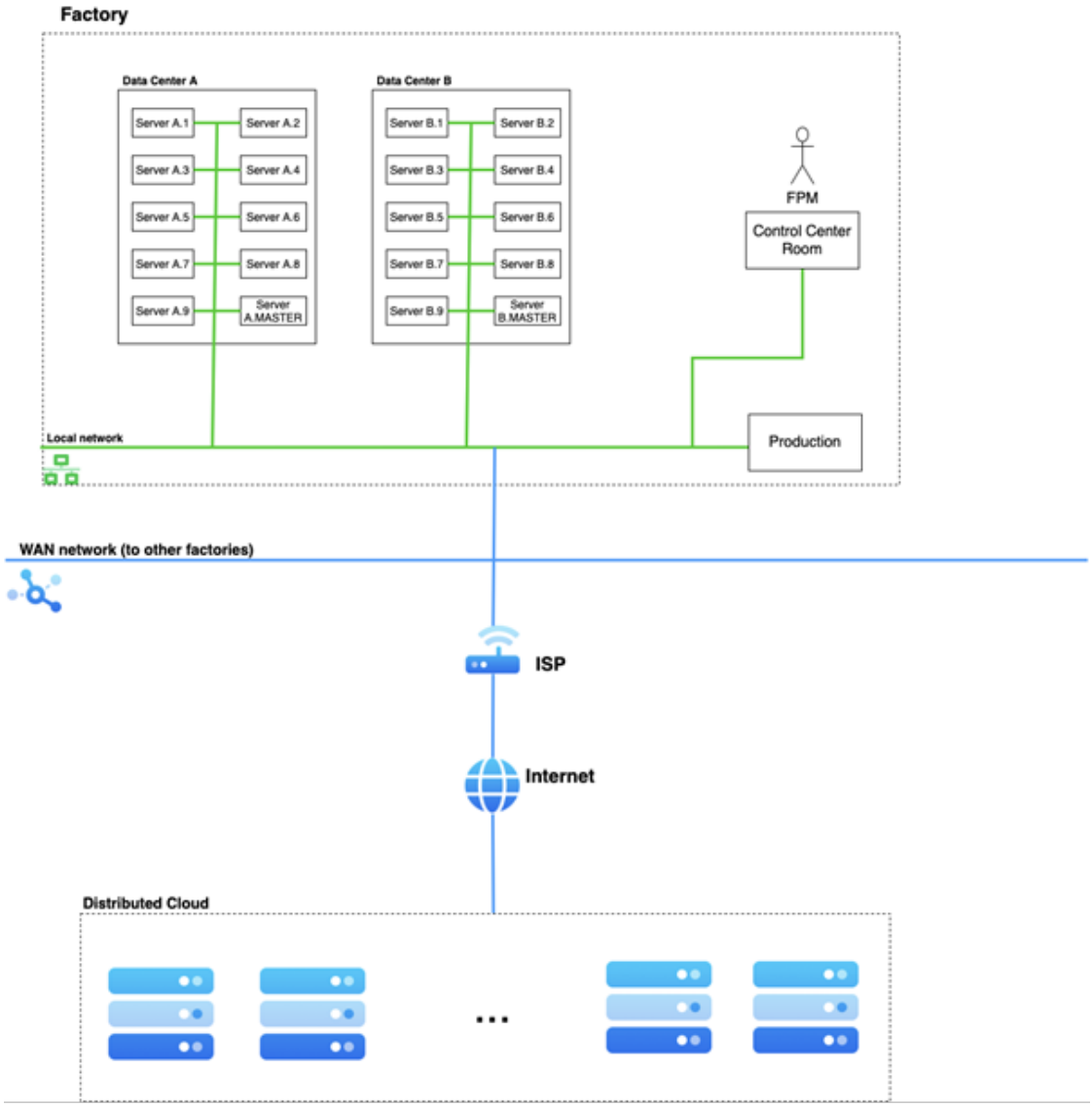
Perciò la spesa totale, ogni 3 anni ammonta a:

$$561600+244224 = 805824 \ USD$$

Inoltre **nel caso di crash dei server** è previsto l'**affitto di un Cloud pay as you go** che al più avrà 16 server, il costo mensile di ognuno dei possibili è di 1354.88 USD, perciò il costo totale nel caso peggiore è di:

$$1254*16=20064 \ USD \text{ al mese}$$

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.

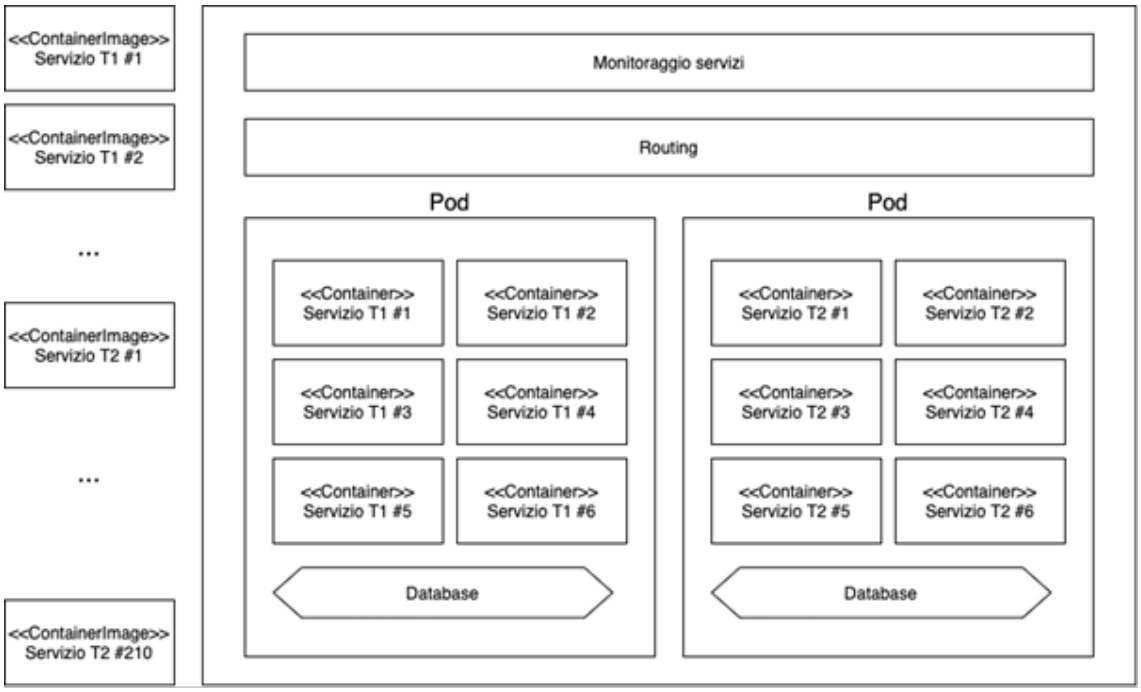
Alla rete locale vengono collegati tutti i server, oltre alla Production, cioè l’insieme delle macchine che si occupano di produrre i chip, e la Control Center Room, una stanza dedicata al controllo dell’andamento della produzione.
La fabbrica è connessa alle altre fabbriche tramite una rete WAN privata, 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

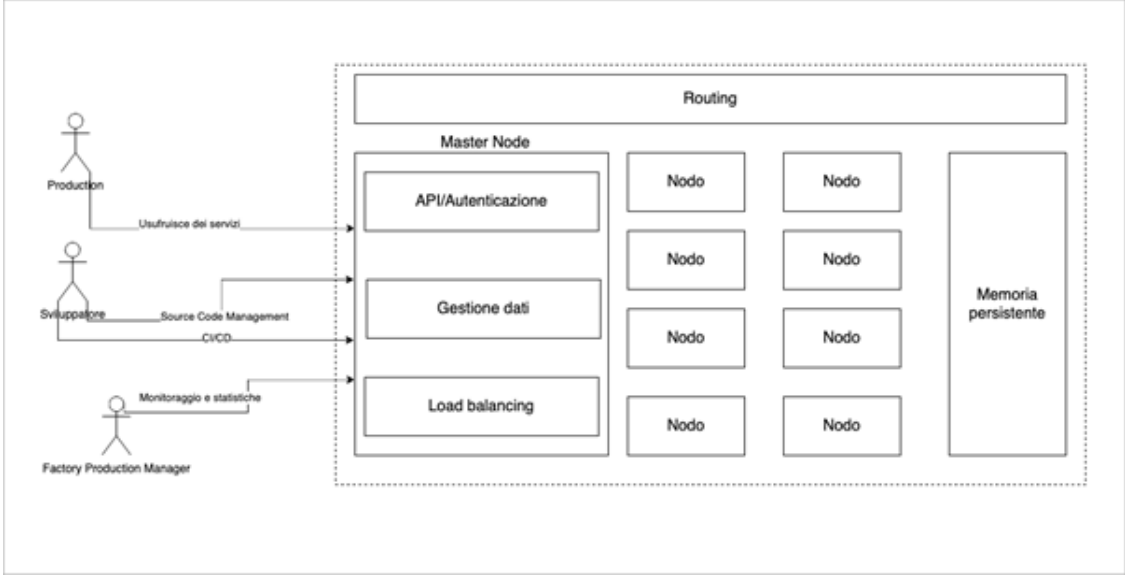
Disponibilità al 99,999%: Viene raggiunta tramite la ridondanza dei data center, poiché ogni server di un data center ha un 100% di utilizzo di CPU medio, e nel caso in cui un data center smetta di funzionare, il carico di lavoro viene momentaneamente spartito tra cloud pay as you go e l’altro data center come precedentemente descritto
.
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.