POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Informatica

Tesi Magistrale

Analisi delle potenzialita' di risparmio energetico ottenibili mediante la condivisione di risorse tra dispositivi IoT

Relatore

Prof. Fulvio RISSO

Candidato

Barbara LI GREGNI

matr. s256782

Dicembre 2021

Alla mia famiglia, ai miei amici, ai miei colleghi e a tutte le persone che mi vogliono bene.
Un pensiero ai miei nonni che mi supportano dall'alto.

Sommario

Oggigiorno l'**ICT** è ovunque - i dispositivi e i servizi "Information and Communication Technology", negli ultimi decenni, hanno profondamente cambiato il modo in con cui gli esseri umani lavorano, viaggiano, giocano e interagiscono. Si stima che almeno l'80% delle famiglie abbiano almeno un PC a casa, il 90% uno smartphone e, in generale siamo circondati da un numero sempre crescente di device IT. Questo trend accresce la domanda di servizi digitali che vengono erogati dai Data Center, spina dorsale di un mondo sempre più digitalizzato. "*ICT everywhere*" ha un impatto crescente sul nostro ambiente; si stima che le emissioni di CO_2 dovute alle tecnologie ICT nel loro insieme rappresentano oltre il 2% delle emissioni globali, e che questo sia un valore destinato ad aumentare. Per questo motivo i consumi dell'ICT sono un tema molto studiato e analizzato dagli esperti. Negli anni i consumi sono stati tenuti sotto controllo dagli efficientamenti `Hardware` e `Software` con stati di basso consumo e altre tecniche. Negli ultimi decenni però il mondo ICT ha subito profonde trasformazioni, influenzando ed essendo influenzato dal cambiamento di paradigma socioeconomico del modello *"ownership-centered"* a quello *"sharing-oriented"*, dove la proprietà del prodotto non è più importante ma ciò che conta è l'accesso al servizio che fornisce. Quello che si è rivelato un vero punto di svolta è stato il passaggio dai server fisici a quelli virtuali con l'affermazione della virtualizzazione prima e poi con l' introduzione della containerizzazione e degli orchestratori di container. Tra questi c'è anche Kubernetes e le sue distribuzioni come **K3s**. I progressi fatti in questo campo, però, hanno sempre avuto un focus aziendale incidendo solo indirettamente sugli utenti "comuni". Negli ultimi anni, il successo del mondo IoT, la massiccia diffusione di dispositivi smart nella vita di tutti i giorni hanno aperto uno scenario diverso, nella quale tutti siamo circondati da tanti device digitali, spesso sottoutilizzati. E' in questo contesto che si inserisce il mio lavoro di tesi: analizzare il potenziale risparmio energetico dovuto alla condivisione di risorse tra dispositivi IoT. Avendo i benefici della *legge di Moore* un limite fisico, l'attenzione si sposta sul software. Infatti sfruttando il fatto che le risorse dei device sono sottoutilizzate e con l'utilizzo di un orchestratore come K3s ci si chiede se, spostando dei task da dispositivi edge poco potenti a device più performanti ci possa essere un risparmio energetico.

Indice

Capitolo 1

Introduzione

La ricerca di più potenza e di strumenti più potenti è sempre andata di pari passo con l'industria e il progresso tecnologico. Questo, in particolar modo, è vero nel mondo ICT, dove in pochi decenni abbiamo assistito ad un salto di qualità anche nel significato stesso di "*possedere un dispositivo elettronico*" in termini di disponibilità, diffusione, dimensioni, capacità e ruolo nella vita quotidiana. Insieme allo sviluppo di Internet e alla crescita della larghezza di banda, questo processo ha anche portato ad un cambiamento di paradigma su come e dove vengono forniti i servizi digitali.

1.1 Dal mainframe al cloud... al "water cycle" del computing

1.1.1 La frammentazione del computing

All'inizio l'adozione di risorse informatiche da parte di un'impresa consisteva in pochi enormi e costosi *mainframe*. Negli ultimi decenni del XX secolo abbiamo visto una tendenza alla democratizzazione nell'accesso e nella disponibilità dei dispositivi IT, principalmente a causa del circolo virtuoso di costante innovazione ingegneristica, prodotti più economici e forte domanda. Fu nel 1965 quando Gordon Moore fece le prime previsioni sull'eccezionale crescita delle capacità dei circuiti integrati che presto sarebbe diventato noto come "*la legge di Moore*" [1]. Ciò ha portato, a metà degli anni '90, ad uno scenario in cui molti server e PC si sono diffusi nei luoghi di lavoro. Il fenomeno è stato enfatizzato dall'applicazione della regola "*un'applicazione per server*", ovvero la buona pratica di eseguire ogni servizio o applicazione su una singola macchina sottoutilizzata, cercando di prevenire problemi di sicurezza e compatibilità. Questo ha portato ad uno spreco di spazio e denaro, sia *diretto* dovuto all'acquisto di hardware / consumo energetico, che *indiretto* dovuto a macchine inattive.

1.1.2 Il Cloud: L'era della virtualizzazione e del cloud computing

Gli anni 2000 sono gli anni del vero cambiamento del modello informatico con la diffusione dell' adozione della virtualizzazione. Vari fattori hanno portato a scegliere la virtualizzazione [2]:

- l'urgenza di razionalizzare le risorse hardware e la sua gestione con la realizzazione di data center e reparti IT, con l'obiettivo di superare la regola *un'applicazione per server*.
- una nuova tendenza nel processo di produzione ha aumentato il numero di core disponibili per CPU, quindi ha reso possibile la parallelizzazione dell'esecuzione di diverse istanze virtuali.
- il desiderio di ottenere sempre più flessibilità.
- la necessità di ridurre il consumo di potenza dei Server.

Il passaggio dai server fisici a quelli virtuali si è rivelato un vero punto di svolta, consentendo di ripensare i data center come a pool di risorse comuni da modellare e ridimensionare in base alle esigenze del momento, "*un enorme computer da magazzino*" [3]. Questo concetto si è evoluto negli anni fino a quello che viene genericamente chiamato "*Cloud Computing*": consumare on-demand un servizio digitale scaricato da qualche parte, con la massima flessibilità.

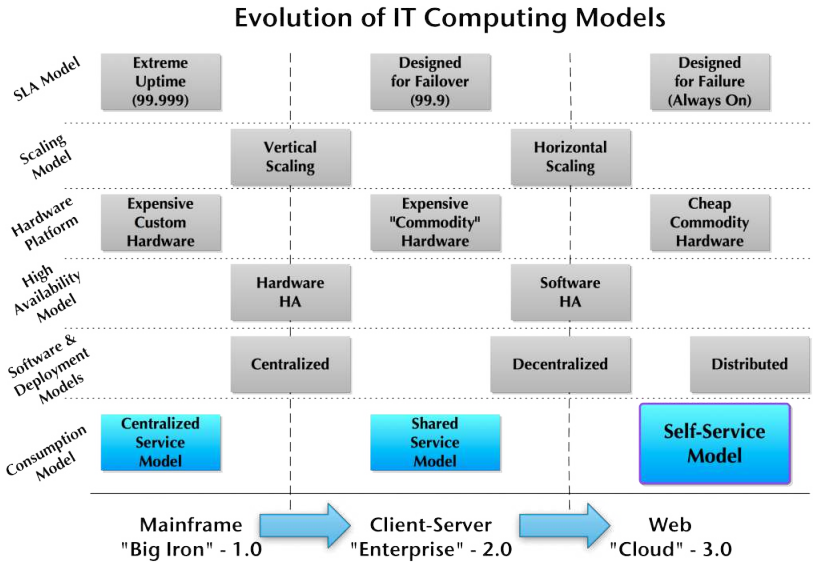


Figura 1.1: Evoluzione dei modelli di computing IT [4]

La definizione di Cloud Computing oggigiorno copre un gran numero di significati in termini di *proprietà*, *tipo di servizio fornito* (ad esempio i modelli *X*-as-a-Service [5]) e *tipo di virtualizzazione* (dalla virtualizzazione "heavyweight" delle macchine virtuali a quella "lightweight" della containerizzazione).

1.2 Il computing "liquido"

Con l'introduzione e la rapida affermazione del progetto *Kubernetes* [6], un orchestratore di Google che mira ad automatizzare e gestire in modo efficiente l'implementazione e il ridimensionamento di applicazioni containerizzate, ha introdotto sostanzialmente un nuovo stack overlay per lo sviluppo e la distribuzione, con un'aggregazione automatica delle risorse nell'astrazione logica di un cluster. Va notato che l'evoluzione dei modelli di calcolo è sempre stata focalizzata al lato aziendale, per la spinta innovativa, la trazione economica, le caratteristiche e la quota di mercato.
Il mercato dei desktop, al contrario, ha sempre tratto beneficio indiretto dai progressi fatti per gli utenti professionali, ad esempio con versioni gratuite e/o limitate del software aziendale. Questo era dovuto principalmente alla mancanza di una quota di mercato significativa (sia dal lato dell'offerta che della domanda):

- Si presuppone che gli utenti comuni non abbiano alcun tipo di background tecnico e sono spesso inclini a consumare servizi digitali *pronti all'uso* con delle interfacce semplici e user-friendly
- C'è solo una piccola frazione di utenti desktop che ha effettivamente bisogno della virtualizzazione e il suo utilizzo è principalmente legato al vantaggio di un'esecuzione parallela di diversi OS.
- Lo scenario medio di un ambiente domestico o di un piccolo ufficio prevede solitamente pochi dispositivi con capacità limitate rispetto a un vero e proprio data center in ambito enterprise.

Nell'ultimo decennio, tuttavia, l'esplosione del mondo IoT, la massiccia diffusione di dispositivi *smart* e la maggiore pervasività dell'IT nella vita quotidiana (e forse anche un po' più di consapevolezza in questo campo) hanno portato ad un scenario diverso, dove ognuno è circondato da un gran numero di dispositivi digitali. Questi device, spesso piccoli e quasi sempre accesi `consumano energia elettrica` e svolgono solo quei pochi compiti per cui sono stati progettati, rimanendo in parte inutilizzati. E' da qui che nasce il concetto di "**computing liquido**", una nuova forma di computing in cui non ci interessa dove viene eseguito il nostro task, l'importante che venga eseguito in sicurezza e con la qualità che ci aspettiamo. Sfruttando l'ambiente Kubernetes (e le sue diverse distribuzioni come K3s) è

possibile modellare un ambiente virtuale unificato in cui le applicazioni possono essere distribuite ed eseguite senza problemi su nodi federati che condividono storage e risorse.

1.3 Obiettivo della tesi

Lo scopo della tesi è quello di analizzare il potenziale risparmio energetico dovuto alla condivisione di risorse tra dispositivi IoT.

I dispositivi IT, dai server ai computer ai dispositivi mobile per fornire i loro servizi vengono alimentati dall'elettricità. Elettricità che viene prodotta da centrali che emettono CO_2 contribuendo al cambiamento climatico. Visti i trend di crescita della diffusione dei dispositivi IoT e di conseguenza dell' aumento del consumo di potenza, diventa necessario trovare una soluzione che permetta di tenere sotto controllo questa tendenza. Sfruttando il fatto che le risorse dei device sono sottoutilizzate e il concetto di *condivisione di risorse* / "*computing liquido*" ci chiediamo se, spostando dei task da dispositivi poco potenti a device più performanti ci possa essere un risparmio energetico.

Dato un certo numero di dispositivi N, alcuni più potenti e altri meno, questi hanno un consumo totale pari a

$$C_{tot1} = \sum_{i=1}^{N} C_i$$

dove C_i è il consumo dei singoli dispositivi.

Utilizzando il paradigma della *condivisione*, avremo che ogni dispositivo avrà un carico aggiuntivo dovuto al software necessario alla condivisione (nel nostro caso **K3s**). Quindi avremo:

$$C_{tot2} = \sum_{i=1}^{N} (C_i + C_{sharing})$$

dove C_i è il consumo dei singoli dispositivi e $C_{sharing}$ è il consumo del software che permette la condivisione di risorse.

L'obiettivo della tesi è quindi quello di capire se C_{tot2} sia minore di C_{tot1} e quindi se effettivamente con un eventuale condivisione di risorse si possa ottenere un risparmio energetico e un miglioramento dell' impatto ambientale. Per poter fare ciò è necessario:

- Misurare i vari C_i (1) verificando se altri studi in letteratura abbiano dei dati puntuali (2) effettuando delle misure sul campo.

- Misurare $C_{sharing}$ del software di condivisione delle risorse(K3s)
- Stimare C_{tot2}
- Confrontare C_{tot1} e C_{tot2}

Capitolo 2

Condivisione di risorse su cluster Kubernates: K3s

In questo capitolo si analizza l'architettura Kubernetes e K3s, mostrandone anche la storia e l'evoluzione nel tempo.[1] Kubernetes (spesso abbreviato in K8s) è un enorme framework ed un esame approfondito richiederebbe molto più tempo, quindi qui forniamo solo una descrizione dei suoi principali concetti e componenti. Ulteriori dettagli possono essere trovati nella documentazione ufficiale [8].

2.1 Kubernates: un po' di storia

Intorno al 2004, Google ha creato il sistema **Borg** [9], un piccolo progetto che all'inizio veniva sviluppato da meno di 5 persone. Il progetto è stato sviluppato in collaborazione con una nuova versione del motore di ricerca di Google. Borg era un sistema interno di gestione di cluster su larga scala, che eseguiva centinaia di migliaia di task di migliaia di applicazioni diverse, provenienti da diversi cluster ciascuno con decine di migliaia di macchine. [9].

Nel 2013 Google ha annunciato **Omega** [10], uno scheduler flessibile e scalabile per grandi cluster di computing. Omega ha fornito un'architettura di scheduler costruita attorno allo stato condiviso, che utilizzava un controllo di concorrenza ottimistico senza lock, al fine di ottenere sia l'estensibilità dell'implementazione che la scalabilità delle prestazioni.

A metà del 2014, Google ha presentato **Kubernetes** come una versione open source di Borg. Kubernetes è stato creato da Joe Beda, Brendan Burns e Craig

[1] Questo capitolo è **in parte** tratto dalla tesi di laurea di un altro membro di questo team di progetto [7]

McLuckie e altri ingegneri di Google. Il suo sviluppo e il suo design sono stati fortemente influenzati da Borg e da molti dei suoi collaboratori iniziali che ci avevano lavorato in precedenza. Il progetto Borg originale è stato scritto in C++, mentre per Kubernetes è stato scelto il linguaggio Go.

Nel 2015 è stato rilasciato Kubernetes v1.0. Insieme al rilascio, Google ha stabilito una partnership con la Linux Foundation per formare la **Cloud Native Computing Foundation** (CNCF) [11]. Da allora, Kubernetes è cresciuta in modo significativo, ed è stato adottato da quasi tutte le grandi aziende. Al giorno d'oggi è di fatto diventato lo standard per l'orchestrazione dei container [12, 13].

2.2 Evoluzione del deployment delle applicazioni

Kubernetes è una piattaforma portatile, estensibile e open source per l'esecuzione e il coordinamento di applicazioni containerizzate su un cluster di macchine. È progettato per gestire l'intero ciclo di vita di applicazioni e servizi utilizzando metodi che forniscono coerenza, scalabilità e alta disponibilità.

Cosa significa "applicazioni containerizzate"? Negli ultimi decenni, l'implementazione delle applicazioni ha visto cambiamenti significativi, illustrati nella figura 2.1.

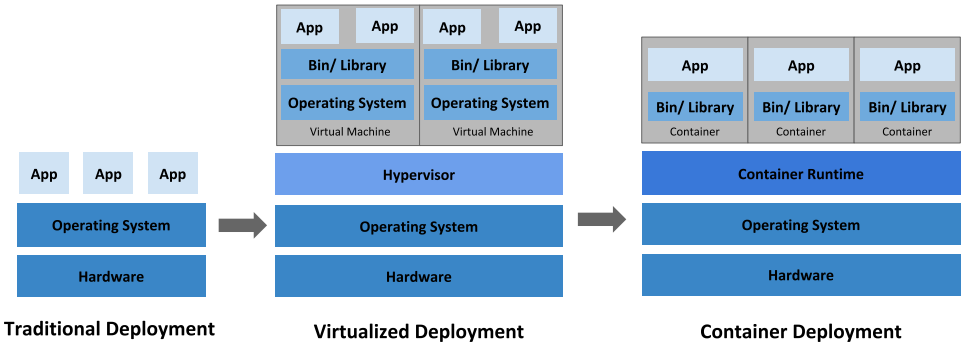


Figura 2.1: Evolution del deployment delle applicazioni

Tradizionalmente, le organizzazioni eseguivano le proprie applicazioni su server fisici. Uno dei problemi di questo approccio era che i limiti delle risorse tra le applicazioni non potevano essere applicati in un server fisico, causando problemi di allocazione delle risorse. Ad esempio, se più applicazioni vengono eseguite su un server fisico, una di esse potrebbe occupare la maggior parte delle risorse e, di conseguenza, le altre applicazioni ne risentirebbero. Una possibilità per risolvere questo problema sarebbe eseguire ogni applicazione su un server fisico diverso, ma chiaramente non è fattibile: la soluzione non potrebbe scalare, porterebbe a un

sottoutilizzo delle risorse e sarebbe molto costoso per le organizzazioni mantenere molti server fisici.

La prima vera soluzione è stata la **virtualizzazione**. La virtualizzazione consente di eseguire più macchine virtuali su un singolo server fisico. Garantisce l'isolamento delle applicazioni tra VM fornendo un elevato livello di sicurezza, poiché le informazioni di un'applicazione non possono essere liberamente accessibili da un'altra applicazione. La virtualizzazione consente un migliore utilizzo delle risorse di un server fisico, migliora la scalabilità, perché un'applicazione può essere aggiunta o aggiornata molto facilmente, riduce i costi dell'hardware e molto altro. Con la virtualizzazione è possibile raggruppare un insieme di risorse fisiche ed esporlo come un cluster di macchine virtuali usa e getta. L'isolamento porta certamente molti vantaggi, ma richiede un sovraccarico piuttosto "pesante": ogni VM è una macchina completa che esegue tutti i componenti, incluso il proprio sistema operativo sopra l'hardware virtualizzato.

Una seconda soluzione proposta di recente è la **containerizzazione**. I container sono simili alle macchine virtuali, ma condividono il sistema operativo con il computer host. Pertanto, i contenitori sono considerati una forma leggera di virtualizzazione. Analogamente a una VM, un container ha il proprio filesystem, CPU, memoria, spazio di processo ecc. Una delle caratteristiche chiave dei container è che sono portabili: poiché sono disaccoppiati dall'infrastruttura sottostante, sono totalmente portabili su cloud e attraverso diverse distribuzioni di sistemi operativi. Questa proprietà è particolarmente rilevante al giorno d'oggi con il "cloud computing": un contenitore può essere facilmente spostato su diverse macchine. Inoltre, essendo "leggeri", i container sono molto più veloci delle macchine virtuali: possono essere avviati, eseguiti e fermati in breve tempo.

2.3 Orchestratore di container

Quando vengono creati centinaia o migliaia di contenitori, diventa essenziale poterli gestirli in modo centralizzato ed efficiente; gli **orchestratori di contenitori** servono proprio a questo scopo. Un orchestratore di container è un sistema progettato per gestire centralmente implementazioni di containerizzazione complesse su più macchine. Come illustrato nella figura 2.2, Kubernetes è di gran lunga l'orchestratore di contenitori più utilizzato. Di seguito forniamo una descrizione di tale sistema.

Kubernetes fornisce molti servizi, tra cui:

- **Discovery dei servizi e bilanciamento del carico** Un contenitore può essere esposto utilizzando il nome DNS o il proprio indirizzo IP. Se il traffico verso un contenitore è elevato, viene fornito un servizio di bilanciamento del carico in grado di distribuire il traffico di rete.

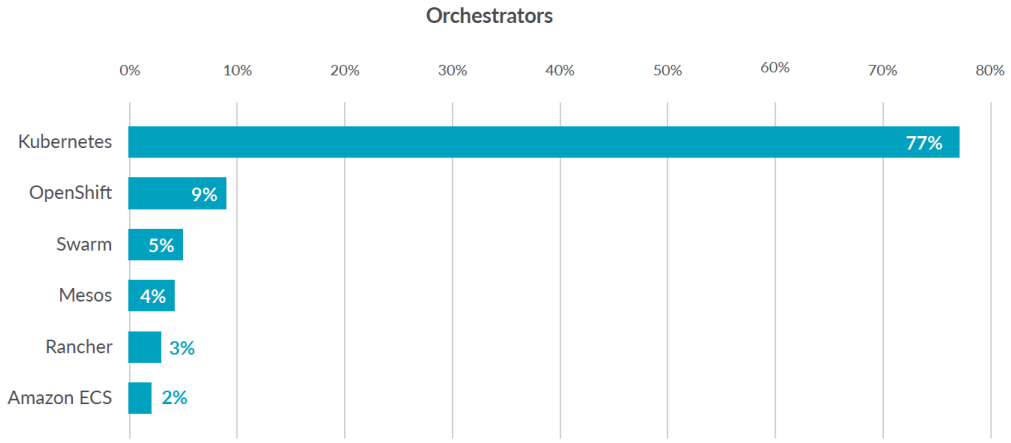


Figura 2.2: Diffusione degli orchestratori di container [14].

- **Orchestramento dell'archiviazione** È possibile montare automaticamente un sistema di archiviazione, ad esempio archivi locali, cloud provider pubblici e altro ancora.

2.4 Architettura

Quando viene fatto il deploy di Kubernetes, viene creato un cluster. Un cluster Kubernetes è costituito da un insieme di macchine, chiamate **nodi**, che eseguono applicazioni containerizzate. Almeno uno dei nodi ospita il "control plane" ed è chiamato **master**. Il suo ruolo è gestire il cluster ed esporre un'interfaccia all'utente. I nodi **worker** ospitano i **pod** che sono i componenti dell'applicazione. Il master gestisce i nodi worker e i pod nel cluster. Negli ambienti di produzione, il control plane di solito viene eseguito su più macchine e un cluster viene eseguito su più nodi, fornendo tolleranza d'errore e alta disponibilità.

La figura 2.3 mostra il diagramma di un cluster Kubernetes con tutti i componenti collegati tra loro.

2.4.1 I componenti del control plane

I componenti del control plane prendono decisioni globali sul cluster (ad esempio lo scheduling), oltre a rilevare e rispondere agli eventi del cluster (ad esempio l'avvio di un nuovo pod). Sebbene possano essere eseguiti su qualsiasi macchina del cluster, per semplicità, in genere vengono eseguiti tutti insieme sulla stessa macchina, che non esegue contenitori utente.

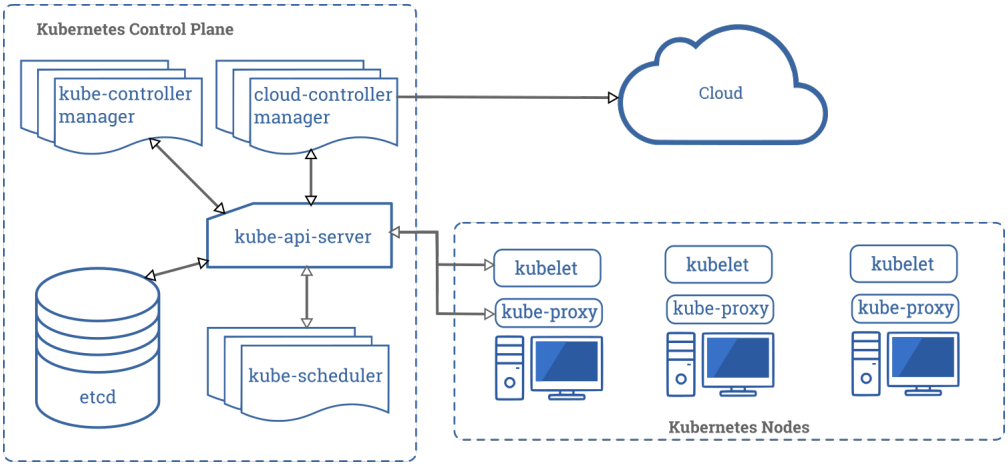


Figura 2.3: Architettura di Kubernetes

Server API

Il **server API** è il componente del control plane Kubernetes che espone l'API REST Kubernetes e costituisce il front-end per il control plane stesso. La sua funzione è intercettare le richieste REST, convalidarle ed elaborarle. L'implementazione principale di un server API Kubernetes è `kube-apiserver`. È progettato per scalare orizzontalmente, il che significa che scala facendo il deploy di più istanze. Inoltre, può essere facilmente ridondato per eseguirne diverse istanze e bilanciare il traffico tra di esse.

etcd

etcd è il componente del control plane che si occupa di mantenere lo stato del sistema, è un database "key-value" distribuito, coerente e ad alta disponibilità utilizzato come "backing store" di Kubernetes per tutti i dati del cluster. Si basa sull'algoritmo di consenso di Raft [15], che consente a macchine diverse di lavorare come un gruppo coerente e sopravvivere alla rottura di uno dei suoi membri. Solo il server API può comunicare con esso.

Scheduler

Lo **scheduler** è il componente del control plane responsabile dell'assegnazione dei pod ai vari nodi. Quello fornito da Kubernetes si chiama `kube-scheduler`, ma può essere personalizzato aggiungendo nuovi scheduler e indicando ai pod di utilizzarli. `kube-scheduler` controlla i pod appena creati che non hanno un nodo assegnato, e dopo averlo identificato glielo assegna. I fattori presi in considerazioni nell'individuare un nodo a cui assegnare l'esecuzione di un Pod ci sono: la richiesta

di risorse del Pod stesso, gli altri workload presenti nel sistema, i vincoli delle hardware/software/policy, le indicazioni di affinity e di anti-affinity, requisiti relativi alla disponibilità di dati, le interferenze tra diversi workload e le scadenze.

kube-controller-manager

kube-controller-manager è il componente del "Control Plane" che esegue e gestisce i processi del controller. Confronta continuamente lo stato desiderato del cluster (dato dalle specifiche degli oggetti) con quello corrente (letto da `etcd`).

Da un punto di vista logico, ogni controller è un processo separato, ma per ridurre la complessità, tutti i principali controller di Kubernetes vengono raggruppati in un unico container ed eseguiti in un singolo processo. Alcuni esempi di controller gestiti dal kube-controller-manager sono:

- `"Controller Node"`: monitora la salute dei nodi e reagisce se uno di questi diventa irraggiungibile
- `"Replication Controller"`: Responsabile del mantenimento del corretto numero di Pod per ogni ReplicaSet presente nel sistema
- `"Endpoint Controller"`: popola gli oggetti Endpoint (che collega Servizi e Pod).
- `"Service Account & Token Controller"`: crea gli account di default e i token di accesso alle API per i nuovi namespace

cloud-controller-manager

Questo componente permette di interagire con i cloud provider sottostante aggiungendo logiche di controllo specifiche per il cloud. Il **cloud-controller-manager** ti permette di collegare il proprio cluster con le API del cloud provider e separa le componenti che interagiscono con la piattaforma cloud dai componenti che interagiscono solamente col cluster. Questo consente al codice del cloud provider e al codice Kubernetes di evolversi indipendentemente l'uno dall'altro. Il `cloud-controller-manager` fornisce a Kubernetes le dipendenze necessarie per ogni cloud provider; nelle release future il codice specifico per ogni cloud vendor verrà mantenuto dal cloud vendor stesso.

I seguenti controller hanno dipendenze verso implementazioni di specifici cloud provider:

- `Node Controller`: controlla che il cloud provider aggiorni o elimini i nodi Kubernetes utilizzando le API cloud.

- `Route Controller`: è responsabile della configurazione delle "network route" nella sottostante infrastruttura cloud
- `Service Controller`: crea, aggiorna ed elimina i "load balancer" del cloud provider
- `Volume Controller`: crea, collega e monta volumi, interagendo con il provider cloud per orchestrarli.

2.4.2 Componenti dei nodi

I componenti del nodo vengono eseguiti su ogni nodo, mantenendo i pod in esecuzione e fornendo l'ambiente di runtime di Kubernetes.

Container Runtime

Il **container runtime** è il software responsabile dell'esecuzione dei container. Kubernetes supporta diversi container runtimes: Docker, containerd, cri-o, rktlet e tutte le implementazioni di Kubernetes CRI (Container Runtime Interface).

kubelet

E' un agente che è eseguito su ogni nodo del cluster. Si assicura che i container siano eseguiti in un pod. Il `kubelet` riceve dal server API le specifiche dei Pod e interagisce con il `container runtime` per eseguirli, monitorando il loro stato e assicurando che i container siano in esecuzione ed integri. La connessione con `container runtime` viene stabilita tramite l'interfaccia di runtime del container ed è basata su gRPC.

kube-proxy

kube-proxy è un agente di rete che viene eseguito su ogni nodo del cluster ed è responsabile della gestione dei Kubernetes Service. Mantiene le regole di networking sui nodi, consentendo la comunicazione di rete con i pod dall'interno o dall'esterno del cluster. Se il sistema operativo fornisce uno strato di filtraggio dei pacchetti, `kube-proxy` lo utilizza, altrimenti inoltra il traffico stesso.

Addons

Addons sono caratteristiche e funzionalità non ancora disponibili nativamente in Kubernetes, ma implementate da pod di terze parti. Alcuni esempi sono DNS, dashboard dotate di "web gui", monitoraggio dei container e logging.

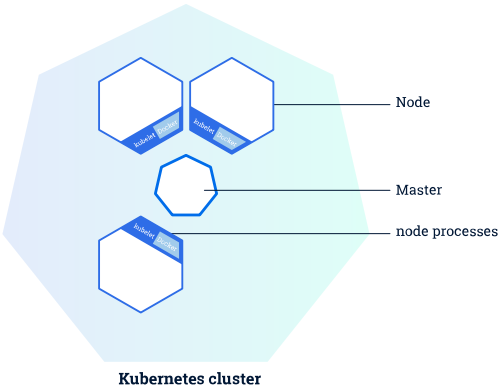


Figura 2.4: Kubernetes master and worker nodes [8].

2.5 Oggetti Kubernetes

Kubernetes definisce diversi tipi di oggetti, che costituiscono gli elementi base. Di solito, un oggetto K8s contiene i seguenti campi [16]:

- `apiVersion`: lo schema versionato di questa rappresentazione dell'oggetto;
- `kind`: un valore che rappresenta la risorsa REST rappresentata da questo oggetto;
- `ObjectMeta`: metadati sull'oggetto, come nome, annotazioni, etichette ecc.;
- `ResourceSpec`: definito dall'utente, descrive lo stato desiderato dell'oggetto;
- `ResourceStatus`: compilato dal server, riporta lo stato attuale della risorsa.

Le operazioni consentite su queste risorse sono le tipiche azioni `CRUD`:

- `Create`: crea la risorsa nel backend di archiviazione; una volta che la risorsa è creata il sistema le applica lo stato desiderato.
- `Read`: viene fornito con 3 varianti
  - `Get`: recupera un oggetto tramite il nome;
  - `List`: recupera tutti gli oggetti di un tipo specifico all'interno di un namespace;
  - `Watch`: trasmette i risultati per uno o più oggetti man mano che vengono aggiornati.

- `Update`: viene fornito con 2 moduli:
  - `Replace`: sostituisce la specifica esistente con quella fornita;
  - `Patch`: applica una modifica a un campo specifico.
- `Delete`: elimina una risorsa;

Di seguito i principali oggetti Kubernetes.

2.5.1 Label & Selector

Le **"Label"** sono coppie chiave-valore associate a un oggetto K8s e vengono utilizzate per organizzare e contrassegnare un sottoinsieme di oggetti.
I **"Selector"** sono le primitive di raggruppamento che consentono di selezionare un insieme di oggetti con la stessa etichetta.

2.5.2 Namespace

I **"Namespace"** sono partizioni virtuali del cluster. Di default, Kubernetes crea 4 namespace:

- `kube-system`: contiene oggetti creati dal sistema K8s, principalmente agent del control plane;
- `default`: contiene oggetti e risorse creati dagli utenti. E' il namespace utilizzato di default;
- `kube-public`: leggibile da tutti (anche da utenti non autenticati), viene utilizzato per scopi come esporre le informazioni pubbliche del cluster;
- `kube-node-lease`: contiene oggetti Lease associati a ciascun nodo. I lease del nodo consentono al kubelet di inviare heartbeat in modo che il Control Plane possa rilevare il failure del nodo.

E' una buona pratica suddividere il cluster in più Namespace per virtualizzare al meglio il cluster stesso.

2.5.3 Pod

I **Pod** sono le unità di elaborazione di base in Kubernetes. Un pod è una raccolta logica di uno o più container che condividono la stessa rete e lo stesso storage. I pod possono contenere più Container ma si preferisce l'utilizzo con uno singolo (a causa del possibile scalamento del pod).

Ci sono dei tipi di container che vengono eseguiti all'interno dello stesso pod. Questi possono essere dei container di supporto al processo principale, chiamati `side-cars`, oppure container di inizializzazione, chiamati `init container`. I pod sono effimeri e non hanno capacità di autoriparazione: per questo motivo sono solitamente gestiti da un controller che gestisce la replica, la tolleranza ai guasti, l'auto-riparazione ecc.

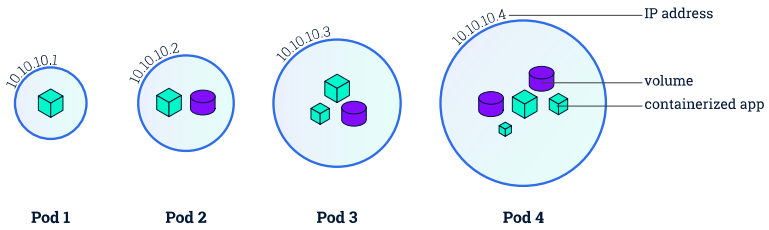


Figura 2.5: I Pod di Kubernetes [8].

2.5.4 ReplicaSet

I **ReplicaSet** controllano un insieme di pod e consentono di ridimensionare il numero di pod che sono attualmente in esecuzione. Se un pod nel set viene eliminato, il ReplicaSet nota che il numero corrente di repliche (letto dallo `Status`) è diverso da quello desiderato (specificato in `Spec`) e crea un nuovo pod. Di solito i ReplicaSet non vengono utilizzati direttamente: Kubernetes fornisce un concetto di livello superiore, chiamato **Deployment**.

2.5.5 Deployment

Deployment sono gli oggetti di K8s che gestiscono la creazione, l'aggiornamento e l'eliminazione dei pod. I pod non sono direttamente lanciati sul cluster anche se sono le unità base. Questi sono di solito gestiti da un livello di astrazione superiore chiamato Deployment. Lo scopo principale del Deployment è dichiarare il numero di repliche da eseguire per un pod. Quando un deployment viene creato sul cluster, vengono automaticamente create e monitorate tutte le repliche definite. Se per qualche motivo un pod non è più disponibile, viene ricreato automaticamente.

2.5.6 Service

I **Service** sono l'astrazione che permette di esporre un'applicazione in esecuzione su un set di pod come servizio di rete. Un Service permette di raggruppare un

set di Pod sotto una singola risorsa, risolvendo il problema del dover conoscere l'indirizzo IP di ognuno. Esistono principalmente quattro tipo di servizi a seconda del *ServiceType*:

- `ClusterIP`: Servizio accessibile solo dall'interno del cluster; è il tipo predefinito;
- `NodePort`: espone il Servizio su una porta statica dell'IP di ogni Nodo; è possibile accedere al Servizio NodePort, dall'esterno del cluster contattando `<NodeIP>:<NodePort>`;
- `LoadBalancer`: espone il Servizio esternamente utilizzando un bilanciatore di carico di un cloud provider;
- `ExternalName`: mappa il Servizio su uno esterno in modo che le app locali possano accedervi.

2.6 K3s

K3s o «RancherK3s», anche noto come "The Lightweight Kubernetes Distribution Built for the Edge" è una distribuzione "alleggerita" di Kubernetes prodotta da Rancher labs, facile da installare e pensata per scenari "resource constrained" (nodi di computing con risorse limitate) e "low touch operations" (in posizioni remote non presidiate, con risorse limitate o all'interno di dispositivi IoT).

Alcuni scenari e ambienti d'impiego particolarmente adatti a K3s sono l'edge computing, le architetture ARM, i device/appliance IoT e la CI (Continuous Integration, nell'ambito dello sviluppo agile del software) [17]. K3s è leggera (impacchettato in un singolo binario di circa 40 MB) ed è molto facile da installare. In questo binario è racchiuso tutto ciò che è necessario per eseguire Kubernetes, incluso il runtime del container e tutte le utility host come iptables.
I requisiti sono molto ridotti: Linux 3.10+, 512 MB di RAM per server, 75 MB di RAM per nodo, 200 MB di spazio su disco, x86_64, ARMv7, ARM64.
K3S raggruppa i componenti di Kubernetes (kube-apiserver, kube-controllermanager, kube-scheduler, kubelet, kube-proxy) in processi combinati che vengono presentati come un semplice modello server e agente. L'esecuzione k3s server avvierà il server Kubernetes e registrerà automaticamente l'host locale come agente. Per aggiungere più nodi al cluster, basta eseguire *"k3s agent –server $URL token $TOKEN"* sull'host che si unirà al cluster.

2.6.1 Architettura K3s

K3S esegue un cluster Kubernetes completo all'edge, il che significa che non è un modello di distribuzione decentralizzato e ogni cluster edge ha una propria

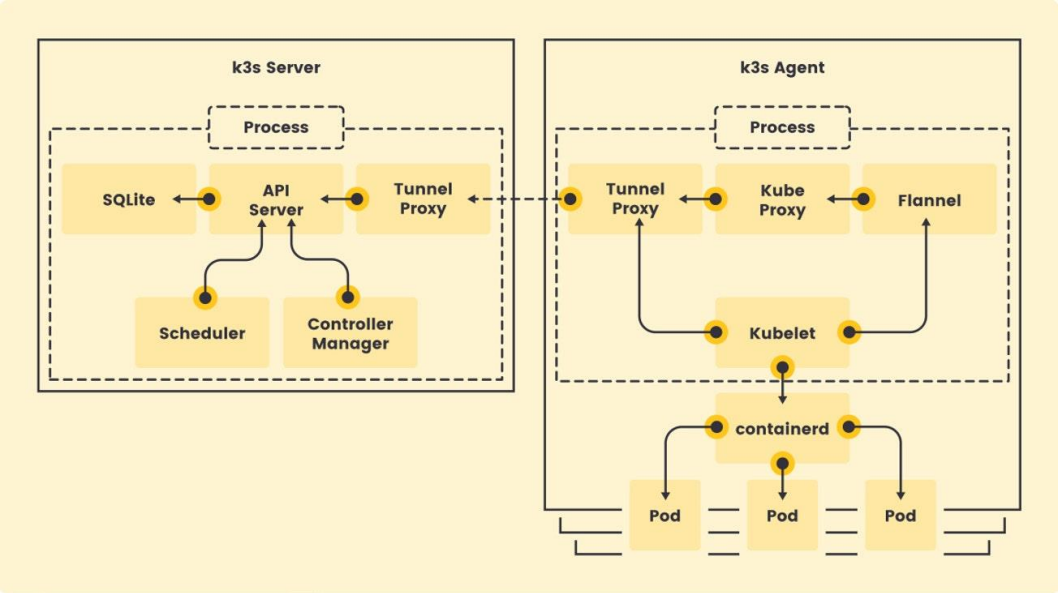


Figura 2.6: Architettura di K3s [18].

logica di controllo. K3s/Rancher costituisce una possibile soluzione al problema dell'interoperabilità tra cluster diversi che, potenzialmente, possono operare anche in maniera autonoma. La presenza di una istanza completa di Kubernetes (o K3s) su ciascun cluster, infatti, garantisce la possibilità che questo possa "auto-orchestrarsi" nell'eventualità in cui il cluster risulti impossibilitato a comunicare con i suoi peer. Di contro, il carico computazionale richiesto su ogni "mini-cluster" potrebbe risultare comunque eccessivo.

Questa soluzione può essere impiegata in un'ambiente multi edge. La soluzione K3s/Rancher consente dunque di gestire cluster medio-grandi all'Edge layer al prezzo di maggior overhead computazionale, richiedendo la presenza di una istanza "master" di Kubernetes in ciascuno di tali cluster. Per cui, può risultare idoneo a scenari in cui è necessario che i singoli cluster siano in grado di operare indipendentemente dalla loro capacità di comunicare con un'istanza "master" remota (ad es. in Cloud layer). Infatti, in assenza di connettività o in condizioni non ottimali di comunicazione, il cluster K3s all'Edge layer risulterebbe comunque dotato di un control-plane che possa adempiere ai task di orchestrazione.

K3s è formato da due processi: il server **"Master Node"** e l'agent **"Working Node"**. Possono essere eseguiti separatamente, ma per impostazione predefinita, K3s esegue un processo che include entrambi. Nella parte seguente una panoramica funzionale di K3s.

K3s Server Node, c.d. K3s Master Node (control-plane) è formato da vari componenti:

- `K3s API Server`: convalida e configura i dati per gli oggetti Pod, servizi, replication controller, ecc. Offre operazioni REST e il front-end dello stato condiviso del cluster;
- `Controller e Scheduler`: componenti base del control plane di Kubernetes
- `SQLite3 DBMS`: datastore di default del cluster K3s, viene usato per disaster recovery. E' costituito da un DB key/value che garantisce coerenza/consistenza (ed eventualmente alta disponibilità) di tutti i metadati relativi allo stato del cluster;
- `K3s agent process`: processo di servizio speciale, utile alle operazioni di join/configurazione iniziali. E' in esecuzione su entrambe le tipologie di Node, Server e Agent. Nel caso del K3s Server Node, agisce come server process in attesa delle connessioni WebSocket aperte dalle controparti (K3s agent process client-side) in esecuzione sui K3s Agent Node.
- `Reverse Tunnel Proxy`: componente che toglie la necessità di comunicazione bidirezionale tra server e agente, il che significa che non è necessario introdurre regole sui firewall affinché i server parlino con gli agent

In questa configurazione base, ogni K3s Agent Node è registrato/autenticato presso lo stesso K3s Server Node. L'amministratore di sistema (K3s user) può manipolare le risorse del cluster K3s invocando il K3s API server (in esecuzione sul K3s Server Node) mediante i comandi usuali necessari a configurare, monitorare e controllare un tipico cluster Kubernetes di risorse orchestrate.

K3s Agent Node, c.d. K3s Worker Node (data-plane) è responsabile dell'esecuzione di kubelet e kube-proxy. Inoltre:

- `K3s agent process`: processo di servizio speciale utile alle operazioni di join/configurazione iniziali, in esecuzione su entrambe le tipologie di Node, Server e Agent. Nel caso dei K3s Agent Node, agisce come client process che apre una connessione WebSocket verso la controparte (K3s agent process server-side) mediante la quale i K3s Agent Node si registrano al K3s Server Node. Tale connessione (client-side) è mantenuta da un load-balancer client-side integrato nel processo K3s agent.
- `Network policy controller`: per far rispettare le "network policies"
- `Load Balancer interno`: che bilancia il carico delle connessioni tra tutti i server API in configurazione HA

- `Flannel`
- `Containerd` come runtime container

2.6.2 Deplyment di K3s

K3s supporta quattro tipi di deployment diversi

- Nodo singolo
- Nodo singolo con più agent
- HA con DB esterno
- HA con DB integrato

[19].

K3s singolo nodo

Il modo più semplice per fare il deploy di k3s è fare un cluster a nodo singolo, che esegue sia il server che l'agent integrato. Questo metodo semplificato è ottimizzato per i dispositivi edge perché basta eseguire il comando curl e distribuirà un cluster Kubernetes completo. Questa soluzione non fornisce HA (Figura 2.7).

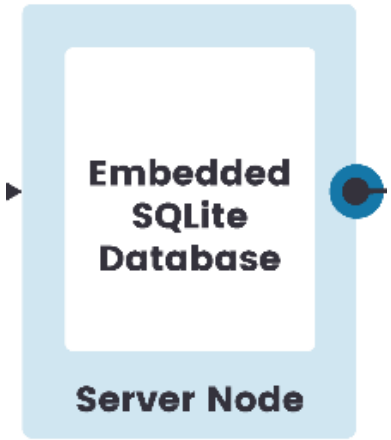


Figura 2.7: Deployment di K3s a nodo singolo.

Nodo singolo con più agent

Anche questo metodo utilizza un singolo server e non fornisce supporto HA. Tuttavia, è possibile aggiungere agent "working node" al cluster. Gli agent si connettono al server in modo sicuro utilizzando un token che il server stesso genera all'avvio. In alternativa, è possibile specificare un token personalizzato all'avvio del server (Figura 2.8).

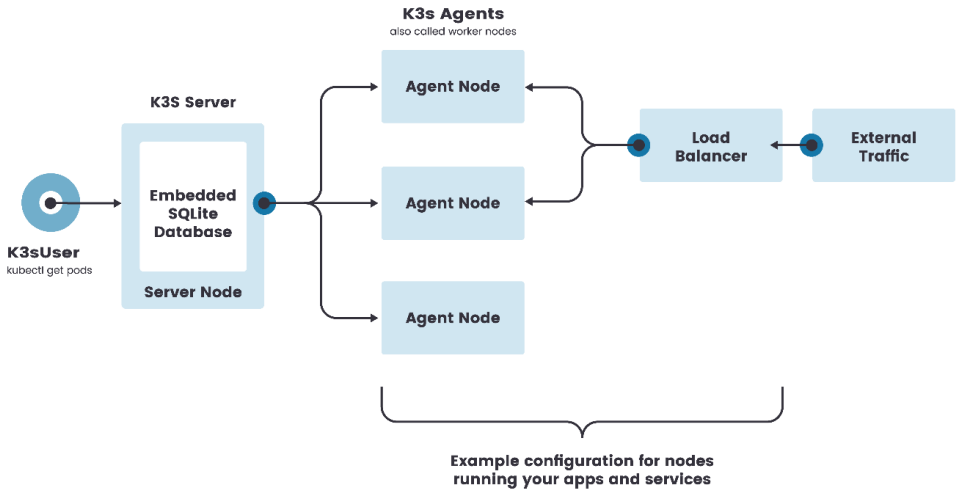


Figura 2.8: Deployment di K3s a nodo singolo con più agent[18].

HA con DB esterno

I cluster a server singoli possono soddisfare una varietà di casi d'uso, ma per gli ambienti in cui il tempo di attività del control plane è fondamentale, è possibile eseguire k3s in configurazione HA (Figura 2.9). Un cluster HA K3 è composto da:

- Due o più nodi server che serviranno l'API Kubernetes ed eseguiranno altri servizi del Control Plane.
- Un datastore esterno (al contrario del datastore SQLite incorporato utilizzato nelle configurazioni a server singolo). E' possibile usare Kine per supportare diversi database esterni, inclusi mysql, postgres e etcd.

I vantaggi di questo metodo sono la disponibilità di diverse opzioni di database e un servizio di load balancing integrato. Inoltre, non è richiesto alcun quorum,

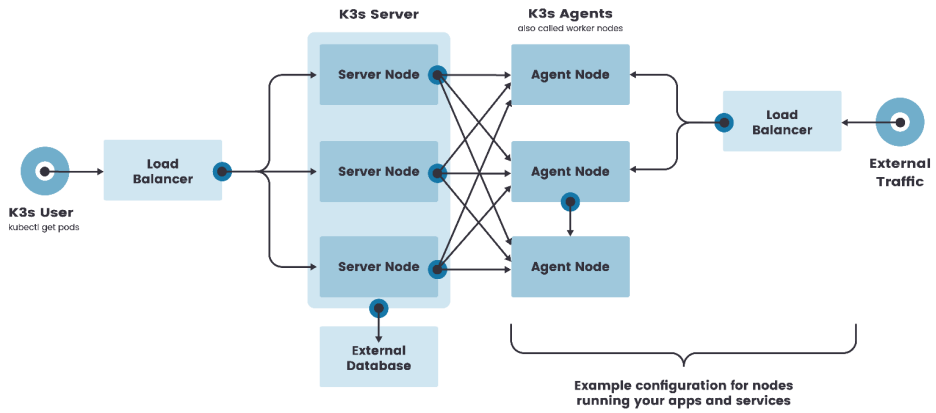


Figura 2.9: Deployment di K3s in HA con DB esterno [18].

poiché l'integrità dei dati è gestita dal DB server. Questo significa che se ho due server e uno si guasta il cluster continuerà a funzionare.

L'unico aspetto negativo di questa architettura è che è necessario configurare manualmente il database esterno, quindi non è realmente ottimizzato per i dispositivi edge. Viene utilizzato più frequentemente in ambienti cloud con un servizio di database gestito come back-end.

HA con DB integrato

Questo metodo utilizza un vero server etcd che è incorporato. Non sono necessari passaggi aggiuntivi per configurare un database. Questo lo rende perfetto per i dispositivi edge. L'unico aspetto negativo di questo metodo è che è basato sul quorum, quindi sono necessari tre server per mantenere il quorum (Figura 2.10).

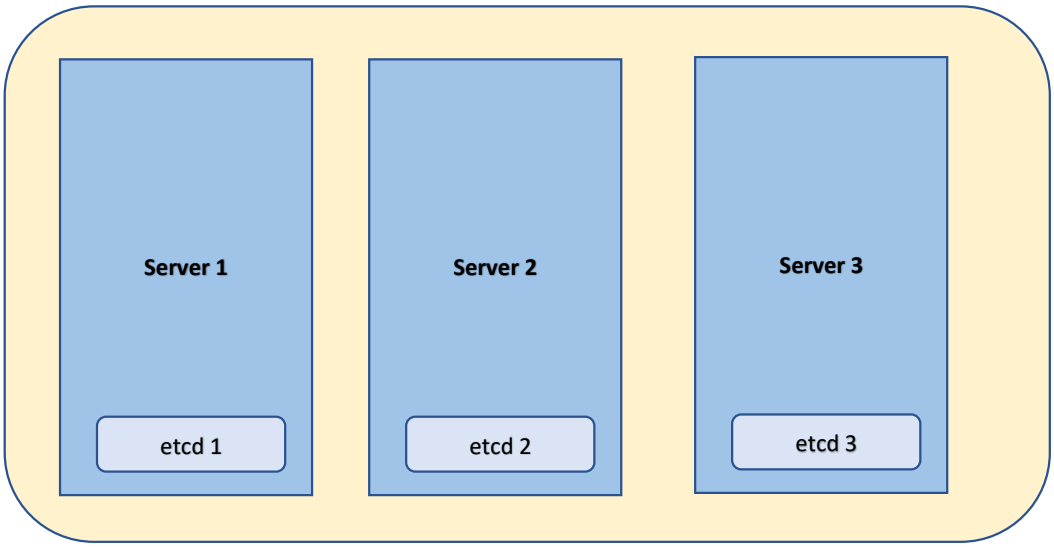


Figura 2.10: Deployment di K3s in HA con DB integrato.

Capitolo 3

Consumo di potenza

3.1 Introduzione

Oggigiorno l'**ICT** è ovunque - i dispositivi e i servizi *"Information and Communication Technology"*, negli ultimi decenni, hanno profondamente cambiato il modo in con cui gli esseri umani lavorano, viaggiano, giocano e interagiscono. Un numero crescente di persone si guadagna da vivere lavorando davanti ad un computer, e molti processi industriali e agricoli devono in qualche modo essere controllati o monitorati da dispositivi intelligenti. Molte auto sono dotate di un GPS che rende più semplice la navigazione su strade sconosciute e stima il tempo di arrivo, anche tenendo conto di fattori real-time come il traffico e i lavori stradali. L'intrattenimento ha sempre più un'impronta digitale, con videogiochi, giochi online e dispositivi intelligenti che permettono anche di allenarsi di fronte ad uno schermo. La forte ascesa dei social come Facebook e la continua proliferazione di smartphone mostra che la comunicazione e l'interazione tra persone avviene sempre più tramite piattaforme digitali. Questo **"ICT everywhere"** ha un impatto, sempre più crescente sul nostro ambiente; effetto che si presenta in molte forme diverse espresse come "footprint". [20] Ad esempio, la produzione e l'utilizzo di apparecchiature ICT hanno associato un'impronta energetica e un'impronta in termini di emissioni di CO_2. L'inquinamento associato all'estrazione di metalli rari per l' ICT, o lo smaltimento di apparecchiature rotte o fuori uso possono essere considerate come parte del "footprint" ambientale. Tra i fattori che influenzano il cambiamento climatico c'è anche il consumo energetico dei vari device ICT. Si stima che le emissioni di CO_2 dovute alle tecnologie ICT nel loro insieme — comprendendo dispositivi digitali personali, reti di telefonia mobile e televisori — rappresentano oltre il 2% delle emissioni globali. Nelle successive sezioni verranno analizzati alcuni aspetti riguardanti l' ICT e il relativo consumo.

3.1.1 Il consumo di potenza

In generale, come mostrato nella figura 3.1 la potenza che viene consumata da qualsiasi sistema consiste di due parti principali [21]:

1. **Consumo energetico statico ($\mathbf{P_{statico}}$)** - SPC:
   Il consumo di energia statica è la potenza consumata dai componenti del sistema. SPC è causata dalle perdite di correnti dei circuiti attivi nel sistema alimentato. È indipendente dalle frequenze di clock e non varia a seconda dei diversi scenari di utilizzo, ma dipende dal tipo di transistor e dalla tecnologia applicata al processore del sistema. La riduzione dell'SPC richiede la riduzione delle perdite di corrente, e questo può essere fatto in tre modi:

   (a) `Ridurre la tensione fornita` per esempio con la rinomata tecnica chiamata *"Supply Voltage Reduction"* (SVR) che viene applicata ai componenti di un sistema (ad es. CPU, memorie cache).

   (b) `Riducendo la dimensione del circuito del sistema`, sia progettando circuiti con meno transistor, o togliendo gli alimentatori ai componenti inattivi per ridurre il numero effettivo di transistor.

   (c) `Raffreddare il sistema applicando tecnologie di raffreddamento`. Le tecnologie di raffreddamento possono ridurre la potenza di dispersione consentendo ai circuiti di funzionare più velocemente sfruttando il fatto che l'elettricità incontra meno resistenza a basse temperature. Questo inoltre elimina alcuni effetti negativi delle alte temperature, in particolare il degrado dell'affidabilità e aumenta l' aspettativa di durata di un chip.

   I tre metodi sopra menzionati richiedono il miglioramento della progettazione del sistema di basso livello, che porta quindi alla riduzione della potenza SPC.

2. **Consumo energetico dinamico ($\mathbf{P_{dinamico}}$)** - DPC:
   Il consumo di energia dinamico deriva dall'utilizzo dei componenti del sistema. Il DPC è determinato dall'attività del circuito e dipende principalmente dalle frequenze di clock, dall' attività di I/O e dallo scenario di utilizzo. Ci sono due fonti di DPC;

   (a) `Capacità commutata (SCDPC)`: SCDPC è la principale fonte di DPC. In questo caso, la potenza consumata è un sottoprodotto di carica e scarica dei condensatori dei circuiti.

   (b) `Corrente cortocircuito (SCCDPC)`: SCCDPC è la fonte minore di DPC. La potenza consumata qui deriva dalla corrente di commutazione tra i transistor.

Quindi, il DPC può essere definito con la seguente espressione:

$$P_{Dinamica} = \alpha C V^2$$

dove α rappresenta l'attività di commutazione nel sistema, C è la capacità fisica, V è la tensione e f è la frequenza di clock della CPU del sistema. I valori di a e C sono determinati in fase di progettazione del sistema.

Il DPC può essere ridotto con quattro metodi:

(a) `Riducendo l'attività di commutazione`.

(b) `Riducendo la capacità fisica che dipende da parametri di progettazione di basso livello come le dimensioni dei transistor`.

(c) `Riducendo la tensione di alimentazione`.

(d) `Riducendo la frequenza di clock`.

La tecnica utilizzata per ridurre DPC è chiamata *Dynamic Voltage and Frequency Scaling* (DVFS), e si basa su una riduzione combinata della tensione di alimentazione e/o frequenza di clock in modo dinamico. Questa tecnica riduce le prestazioni della CPU diminuendo la tensione e la frequenza della CPU quando non è pienamente utilizzata.

La potenza totale P_{tot} consumata in un sistema è quindi:

$$P_{tot} = P_{statica} + P_{dinamica}$$

3.2 Stato dell' arte

3.2.1 Data Center

Panoramica generale

I **Data Center** rappresentano la spina dorsale di un mondo sempre più digitalizzato. La domanda per i loro servizi è cresciuta rapidamente, e tecnologie "data-intensive" come l'intelligenza artificiale (AI), sistemi energetici smart, sistemi distribuiti di produzione e veicoli autonomi promettono di aumentare ulteriormente la domanda.[22] Dato che la domanda dei servizi dei data center aumenta rapidamente, così anche il loro `consumo energetico globale` è in crescita. Ad oggi si stima che i data center consumino circa l'1% del consumo mondiale di elettricità. Alcune analisi affermano che l'energia utilizzata dai data center nel mondo è raddoppiata negli ultimi dieci anni e che triplicherà o addirittura quadruplicherà entro il prossimo decennio. Queste tendenze hanno chiare implicazioni sulla domanda globale

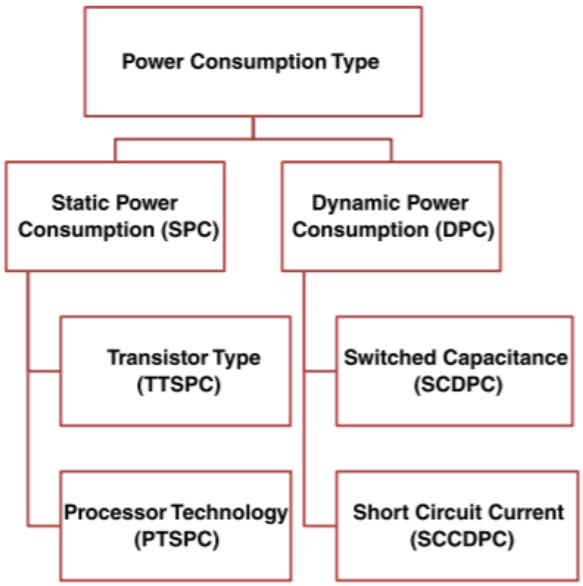


Figura 3.1: Tipi di consumo di potenza: Statico e Dinamico [21].

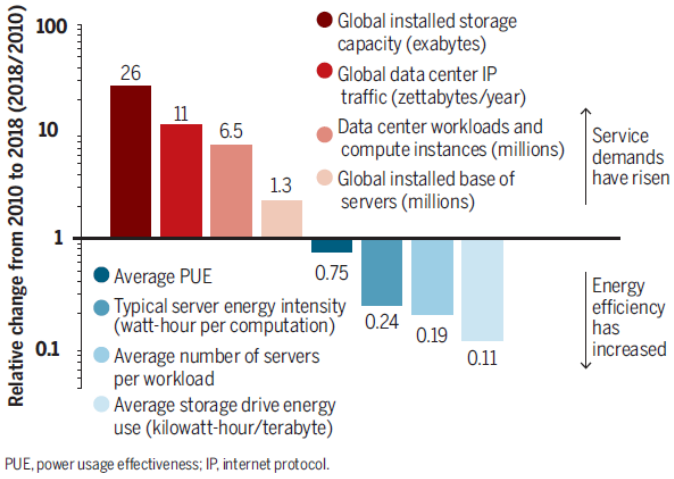


Figura 3.2: Trend del consumo energetico dei data center globali [22].

di energia, ma non tengono in considerazione l' `efficientamento energetico` avvenuto in parallelo (Figura 3.2).

Integrando questi dati provenienti da diversi fonti vediamo una più modesta

crescita di consumo a livello globale (Figura 3.3).

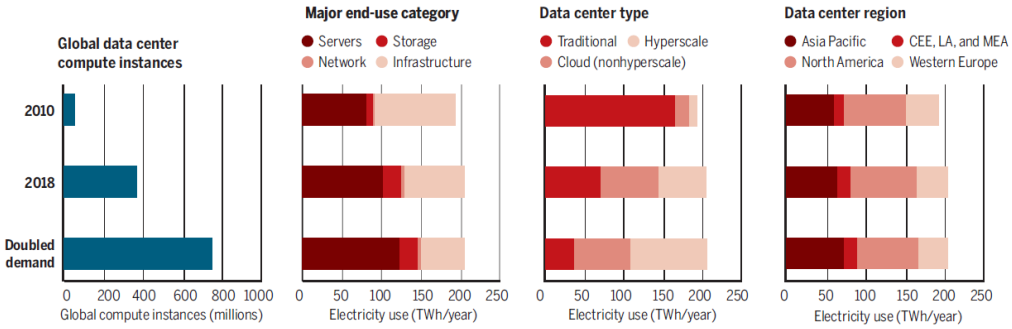


Figura 3.3: Consumo energetico storico e proiezione del consumo energetico previsto con domanda di elaborazione raddoppiata [22].

Recenti analisi stimano che l'uso di energia dei data center è cresciuto da 153 terawattora (TWh) nel 2005 ad un valore compreso tra 203 e 273 TWh nel 2010, per un totale 1,1-1,5% del consumo globale di elettricità.

Il traffico Internet globale è aumentato di quasi il 40% tra febbraio e metà aprile 2020 durante il culmine delle misure di contenimento del Covid-19, trainato dalla crescita dello streaming video, delle videoconferenze, dei giochi online e dei social network. Questa crescita si aggiunge alla crescente domanda di servizi digitali nell'ultimo decennio: dal 2010, il numero di utenti Internet in tutto il mondo è raddoppiato mentre il traffico Internet globale è cresciuto di 12 volte, ovvero circa il 30% all'anno [23]. Si prevede che la domanda di dati e servizi digitali continuerà la sua crescita esponenziale nei prossimi anni, con il traffico Internet globale che dovrebbe raddoppiare entro il 2022 a 4,2 zettabyte all'anno (4,2 trilioni di gigabyte). Si stima che il numero di utenti di Internet da mobile aumenterà da 3,8 miliardi nel 2019 a 5 miliardi entro il 2025, mentre si prevede che il numero di connessioni dell' `Internet of Things` (IoT) raddoppierà da 12 a 25 miliardi. Queste tendenze stanno guidando una crescita esponenziale di richieste ai data center e servizi di rete.

Efficientamenti

Tutto ciò farebbe pensare quindi ad un aumento considerevole dei consumi. Ma dal 2010 il consumo di elettricità per un tipico server è diminuito di un fattore quattro grazie ai miglioramenti dell'efficienza del processore e la riduzione dell' "idle power". Allo stesso tempo, i watt per terabyte di spazio di archiviazione è diminuito di un fattore nove a causa della densità e dall'efficienza delle unità di archiviazione.

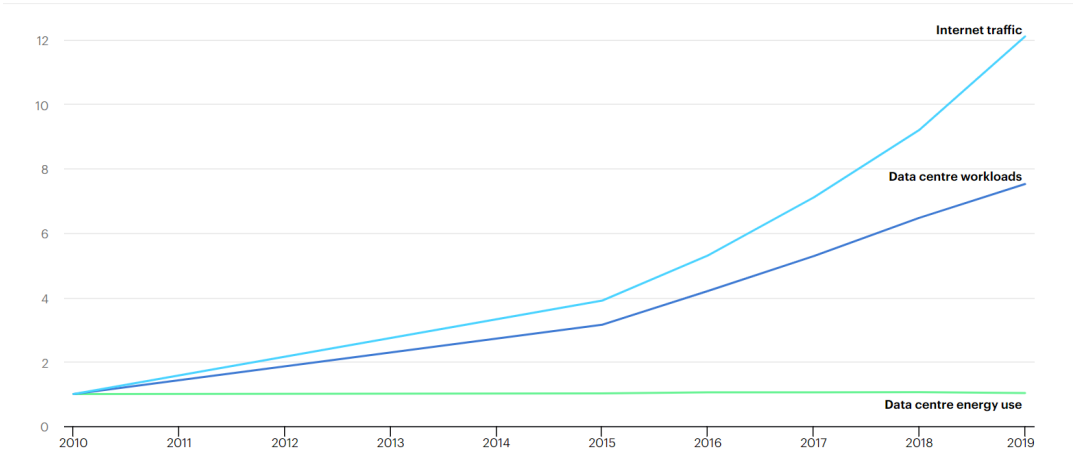


Figura 3.4: Tendenze globali del traffico Internet, dei workload e del consumo energetico dei data center (2010-2019) [23]

Anche le tecnologie delle reti di trasmissione dati stanno rapidamente diventando più efficienti: l'intensità energetica della rete fissa si è dimezzata ogni due anni dal 2000 e l'efficienza energetica della rete mobile (che rappresentano i due terzi del consumo totale delle reti) è migliorata del 10-30% annuo negli ultimi anni grazie alle reti 4G che sono circa cinque volte più efficienti dal punto di vista energetico del 3G e 50 volte più efficienti del 2G.

Un altro aspetto che ha notevolmente rallentato i consumi dei data center è la `virtualizzazione`: la crescita del numero medio di istanze ospitate per server è aumentato di cinque volte con un efficientamento visibile nella figura 3.5.

Accanto a ciò c'è la costante riduzione dell' efficienza del consumo energetico dei DC misurato in `PUE "Power usage effectiveness"`. PUE è un parametro che rende l'idea di quanta potenza elettrica sia dedicata all'alimentazioni degli apparati IT rispetto ai servizi ausiliari come il condizionamento o il consumo degli UPS. Il PUE è il rapporto tra la potenza totale assorbita dal data center (P_{tot}) e quella usata dai soli apparati IT (P_{IT}).

$$PUE = P_{tot}/P_{IT}$$

I tipici data center aziendali progettati con un'infrastruttura obsoleta hanno un PUE nell'intervallo compreso tra due e tre. Questo significa che per un watt di potenza utilizzato per far funzionare le apparecchiature IT, da uno a due watt di potenza sono utilizzato per alimentare e raffreddare le apparecchiature stesse. Per gli standard moderni, questo è altamente inefficiente. La Figura 3.6 illustra

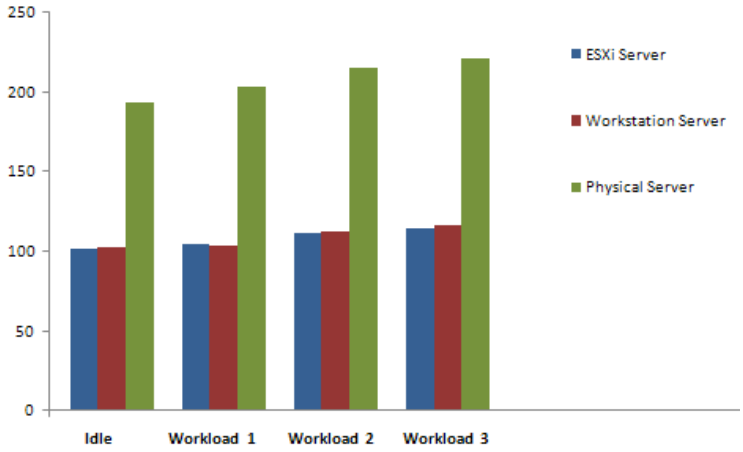


Figura 3.5: Consumi di server fisici e virtuali a confronto. A tutti i workload i server fisici sono evidentemente inefficienti con un consumo quasi doppio rispetto ai server virtualizzati. [24].

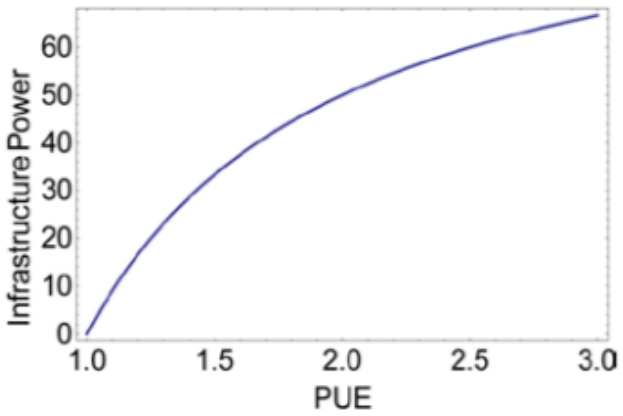


Figura 3.6: Frazione della potenza totale usata dall'infrastruttura del data center in funzione del PUE [25].

la relazione inversa tra l' infrastruttura di un data center e il valore PUE. Per un PUE uguale a 2.0, viene utilizzato il 50% della potenza del DC per finalità non computazionali. Alcuni data center altamente inefficienti possono operare ad un PUE maggiore di 3. Quando il valore PUE aumenta oltre 2.0, viene utilizzato oltre il 50% della potenza del data center per riscaldamento e condizionamento.

Negli anni le stime annuali del PUE hanno evidenziato una chiara tendenza al

ribasso [26], come mostrato nel grafico 3.7

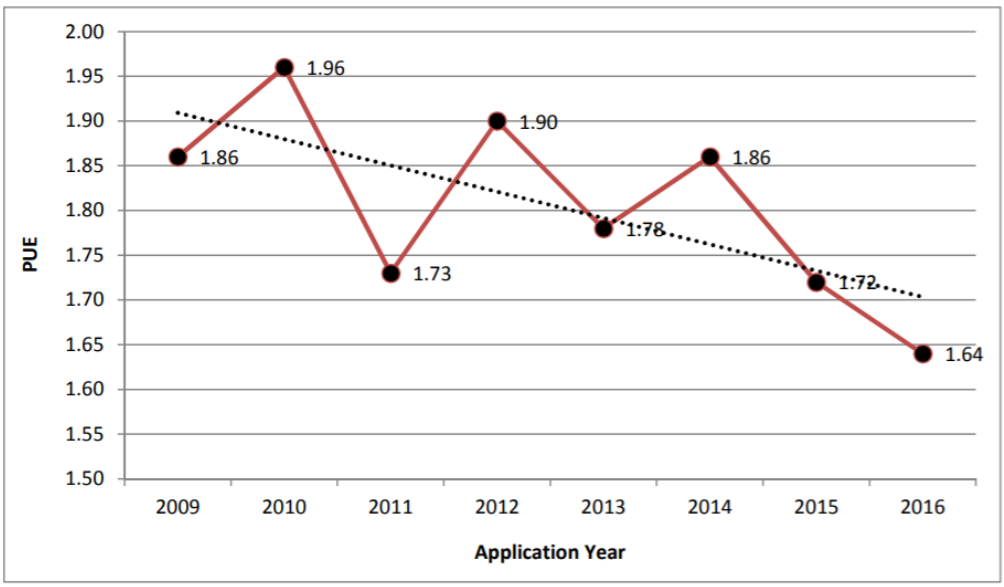


Figura 3.7: Trend PUE negli anni 2009 - 2016. Si nota un progressivo miglioramento dell' indicatore e quindi dell'efficienza dei Data Center [26].

Queste tendenze sono state in gran parte guidate dallo spostando delle istanze di calcolo verso `data center "hyperscale"` che utilizzano sistemi di raffreddamento ultra efficienti e altre importanti pratiche di efficienza per ridurre al minimo il consumo di energia. Nella figura 3.8 possiamo notare come, nonostante la crescita della domanda computazionale, il consumo elettrico sia rimasto pressochè invariato grazie alla migrazione da DC tradizionale a cloud più efficienti.

Consumo di un DC e tecniche

Un Data Center, come osservabile dalla figura 3.9 è composto da due livelli principali: `software` e `hardware`.

L'energia consumata da un data center può di conseguenza, essere suddivisa in due parti [29]:

1. **Hardware**
   a sua volta divisibile in:

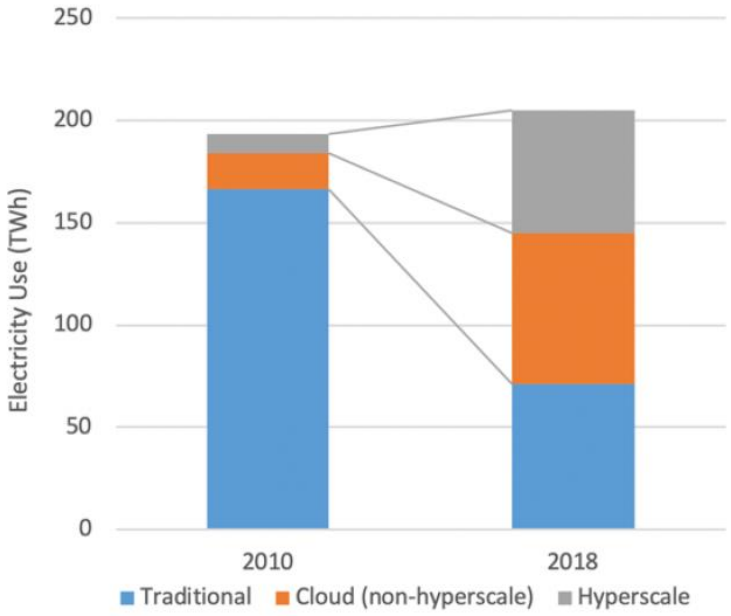


Figura 3.8: Stima del consumo di elettricità dei DC globali per tipo (2010 - 2018). Nonostante l'aumento esponenziale della domanda di servizi dei DC, grazie allo spostamento delle istanze di calcolo da DC tradizionali verso i più efficienti Cloud / Hyerscale, i consumi sono aumentati solo leggermente. [27].

   (a) `apparecchiature IT`: consumo energetico di server, reti, storage, ecc. La modellazione IT si riferisce principalmente ai server e ai loro componenti: processore, memoria, hard disk, ventole e interfaccia di rete.

   (b) `infrastrutture`: consumo energetico da parte delle infrastrutture, ad es. sistemi di raffreddamento e condizionamento.

2. **Software**:
   La modellazione del software si concentra sulla virtualizzazione e sul funzionamento dei sistemi, applicazioni "data-intensive", comunicazioni e tecniche di intelligenza artificiale.

La quantità di energia consumata da questi due sottocomponenti dipende da come è progettato il data center e dall'efficienza dell'apparecchiatura utilizzata.

Nella figura 3.10 possiamo notare che l'infrastruttura di raffreddamento è la parte più energivora di un DC, seguita da server e storage (le percentuali potrebbero differire da data center a data center ma la gerarchia di consumo rimane la stessa).

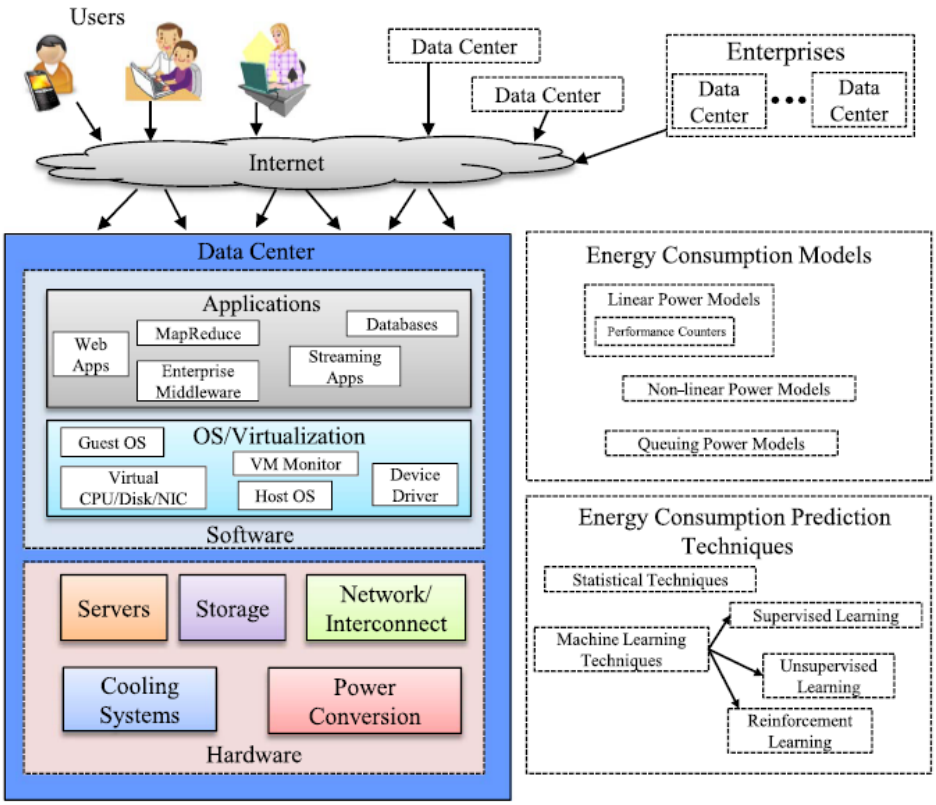


Figura 3.9: Una visione olistica di un DC con una classificazione generale della modellazione e previsione del consumo energetico. I componenti di un data center possono essere classificati in due livelli principali: *software* e *hardware*. [28].

Negli ultimi anni, i ricercatori hanno proposto diverse tecniche per migliorare il PUE e di conseguenza l'efficienza energetica dei DC [30].
Una prima classificazione di queste tecniche è:

- `Tecniche basate su DVFS`:
  Dynamic Voltage and Frequency Scaling (DVFS) è una tecnica di gestione dell'energia ampiamente utilizzata in cui la frequenza di clock di un processore viene regolata dinamicamente per consentire una corrispondente riduzione della tensione di alimentazione per ottenere un risparmio energetico. DVFS è particolarmente utile per i carichi di lavoro legati alla memoria.

- `Tecniche basate sul consolidamento dei server e sulle transizioni`

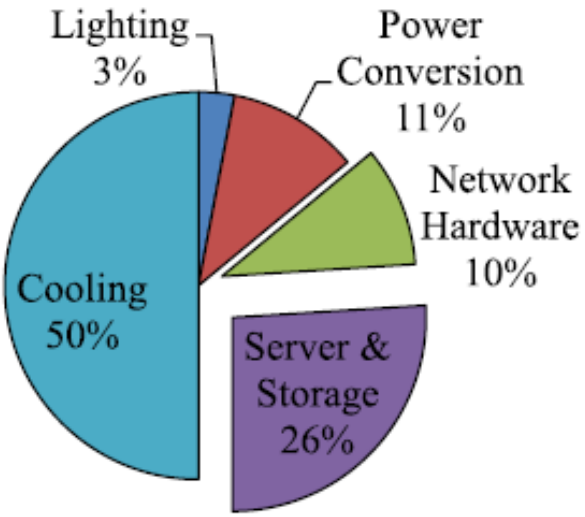


Figura 3.10: Una ripartizione del consumo di energia dei diversi componenti di un data center. [29].

`di stato dell' alimentazione del server (stato di bassa potenza o spento`)
I server moderni in genere operano a bassi livelli di utilizzo, mediamente tra il 10% e il 50% del loro massimo possibile utilizzo (Figura 3.11).

Inoltre, per far fronte ai picchi di domanda e soddisfare le SLA, vengono allocati un'elevata quantità di risorse. Ciò porta ad una scarsa efficienza energetica. Per affrontare questa sfida, sono stati proposti diversi approcci. Il *consolidamento* dei server è uno di questi. E' finalizzato a garantire un utilizzo efficiente delle risorse riducendo il numero totale di server richiesti da un data center, pur continuando a soddisfare la stessa portata di richieste. In questo approccio, le applicazioni esistenti vengono consolidate su un numero inferiore di server, in modo tale che i server inutilizzati possano passare ad uno stato di basso consumo (o addirittura allo spegnimento) e gli altri server possono sfruttare a pieno le loro risorse.
Un altro approccio possibile è la transizione delle risorse del server in modalità a *basso consumo* durante i periodi di bassa attività.

- `Tecniche basate sulla gestione del carico di lavoro o sulla pianificazione`

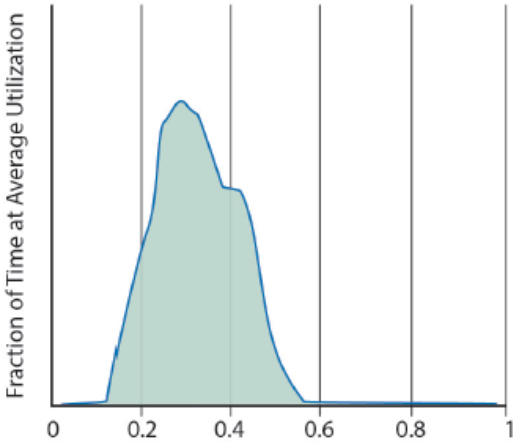


Figura 3.11: Distribuzione media dell'attività di un cluster Google contenente oltre 20.000 server, in un periodo di 3 mesi (gennaio-marzo 2013). [31]

`delle attività`:
I Data Center moderni in genere sono composti da un gran numero di server e, quindi, la decisione sul posizionamento dei carichi di lavoro sui server influisce in modo significativo sulla dissipazione del calore e sul consumo di energia. Ad una cattiva collocazione del workload può conseguire un notevole aumento della temperatura dell'edificio che, a sua volta, aumenterà ulteriormente la dissipazione del calore dei server e quindi incrementerà anche i requisiti di raffreddamento.

- `Tecniche termiche o di gestione termica`:
  Diverse tecniche di gestione dell'energia funzionano in modo inconsapevole dal punto di vista termico, cioè non prendono in considerazione la dipendenza della temperatura con il consumo energetico del server. Tuttavia, la dissipazione di calore dei componenti del processore hanno una forte dipendenza con la temperatura e quindi un aumento della temperatura di esercizio porta ad una maggiore dissipazione del calore, che aumenta ulteriormente la temperatura e così via. Un esempio di quanto asserito è visibile nella figura 3.12,

- `Tecniche per il raffreddamento del data center`
  Tra questi metodi abbiamo [31]:

  - `Gestione attenta del flusso d'aria`
    separare l'aria calda espulsa dai server dall'aria fredda e mantenere il

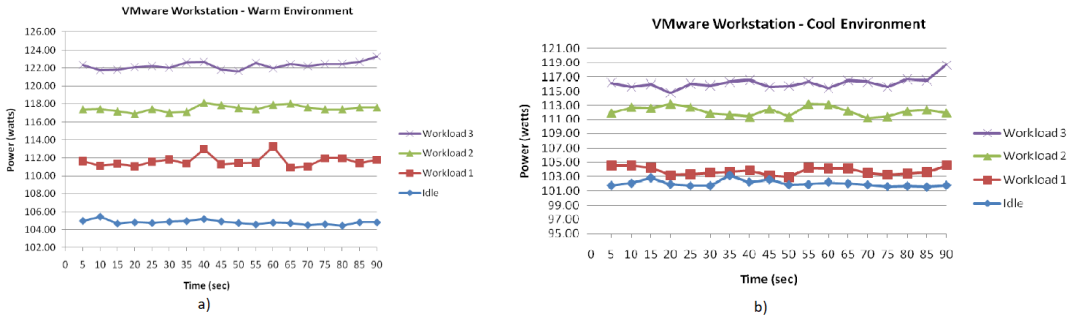


Figura 3.12: Misurazione dei consumi di una Workstation VmWare a temperature diverse; il sistema consuma più potenza con temperature "alte" rispetto a quelle più basse. [32]

percorso verso la serpentina di raffreddamento il più breve possibile in modo che venga spesa poca energia per spostare l'aria fredda o calda per lunghe distanze.

- `"Freecooling"`
  Nei climi più moderati, il "freecooling" può eliminare del tutto o in parte i refrigeratori. Posizionare semplicemente i data center in climi freddi e soffiarvi l'aria esterna è diventata una soluzione molto popolare.
- `Migliore architettura del sistema di alimentazione`
  i consumi dovuti agli UPS e alle perdite di distribuzione dell'alimentazione possono essere considerevoli ridotti utilizzando tecniche più efficienti.

Sulla base di altre caratteristiche/parametri, le tecniche possono essere ulteriormente classificate. La maggior parte delle tecniche mirano a ridurre la potenza media (energia), altre invece mirano a ridurre il consumo di picco o a limitare la potenza (`"power capping"`). La necessità di gestire la potenza di picco è ben compresa e, soprattutto i server, vengono forniti con meccanismi di "power capping" che consentono di limitare il consumo di punta ad una soglia prefissata. Gli sprechi possono essere evitati coordinando più server. Ad esempio, quando i server in un cluster funzionano ad un carico inferiore, la potenza rimasta inutilizzata potrebbe essere utilizzata da altri server per operare a livelli di potenza più elevati rispetto a quanto sarebbe consentito dal loro limite statico. Piuttosto che forzare un livello di potenza inferiore in ogni momento, sono stati studiati metodi che coordinano i limiti di potenza su più server dinamicamente [33]. Nell'implementare queste tecniche però, bisogna tenere in considerazione due proprietà molto importanti:

1. `Velocità`
   Il consumo di potenza di un DC può variare molto rapidamente. Questo a

causa del cambiamento del workload dei server dovuto a volumi di richieste diversi, attività intensive come il backup dei dati, attività in background del sistema operativo come uno scrub del disco o scansioni antivirus o altri problemi come il riavvio simultaneo di server. I metodi di "power capping" hanno una latenza di attuazione dei meccanismi stessi. I meccanismi di limitazione dell'alimentazione a livello di server, sono tipicamente implementati nel firmware della scheda madre del server. Per cambiare la frequenza del processore utilizzano la tensione dinamica e il ridimensionamento della frequenza (DVFS) fino a quando il consumo di energia scende al di sotto del valore desiderato. Questi metodi locali sono molto veloci, in genere limitano la potenza in pochi millisecondi. Tuttavia, la velocità di "capping" può diventare un problema per i metodi di limitazione della potenza coordinati che regolano dinamicamente i limiti su migliaia o decine di migliaia di server. Nella figura 3.13 viene mostrata la latenza totale che serve per ottenere il livello di potenza totale desiderato di un data center.

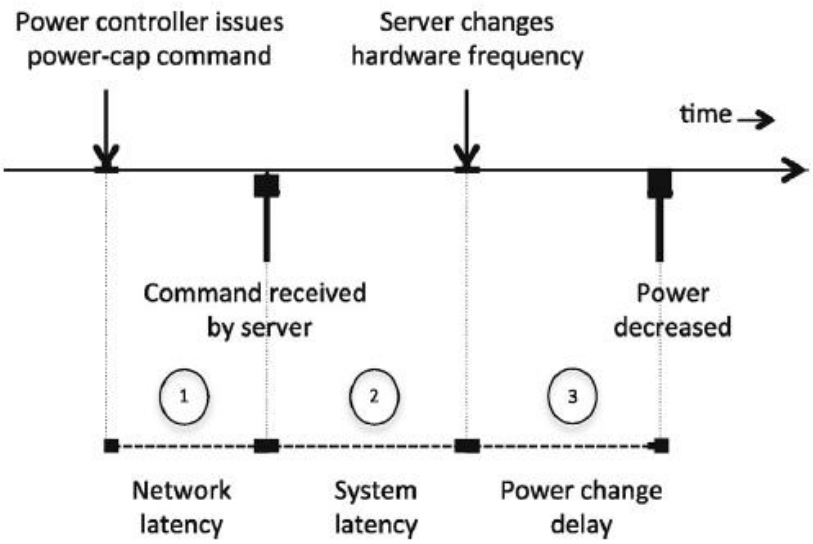


Figura 3.13: Timeline che mostra la latenza per una soluzione di "power capping" coordinato [33]

2. `Stabilità`
   Per alcuni utilizzi (es. server che ospitano servizi online) il sistema potrebbe diventare instabile se l'alimentazione viene interrotta. Una piccola riduzione in potenza ottenuta attraverso i metodi di "power capping" può causare un

aumento incontrollato della latenza dell'applicazione e potrebbe persino ridurre il "throughput" a zero.

Diverse tecniche utilizzano un approccio analitico e offrono algoritmi teorici di controllo che danno garanzie dimostrabili, mentre la maggior parte delle altre tecniche utilizzano un approccio di sistema e si concentrano solo sull' implementazione. Alcuni ricercatori propongono l'uso di fonti energetiche rinnovabili. Alcuni ricercatori mirano a ridurre l'energia del disco, mentre altri si concentrano sul risparmio energetico della memoria principale.

Emissioni CO_2

Questo notevole consumo di elettricità dei data center, e il suo potenziale aumento, solleva preoccupazioni anche per quanto riguarda le *emissioni di anidride carbonica (CO_2)* e quindi al *cambiamento climatico*. Al momento non è ancora possibile stimare con precisione le emissioni totali di CO_2, a causa della mancanza di dati sulla stragrande maggioranza dei data center globali e delle intensità di emissioni (misurate in grammi di CO_2 per kilowattora) delle loro effettive fonti di elettricità . Poche aziende, tra cui Google, Apple, Switch e Facebook, hanno pubblicato tali dati, indicando una tendenza all' approvvigionamento di energia rinnovabile (Figura 3.14).

Si stima che i data center contribuiscono per circa lo 0,3% alle emissioni complessive di carbonio. Questo mette il carbonio dell'ICT alla pari delle emissioni da carburante dell'industria aereonautica [34].

Modelli

Per studiare il consumo di potenza viene utilizzata la **Modellazione statistica** [35], una metodologia semplificata e matematicamente formalizzata per approssimare la realtà e fare previsioni a partire da queste approssimazioni. Il modello statistico è caratterizzato da una o più equazioni matematiche che rappresentano il modello stesso. Le tecniche utilizzate per modellare il consumo statisticamente possono essere generalmente suddivise in due categorie, "top-down" e "bottom-up". La terminologia fa riferimento alla posizione gerarchica di input dei dati rispetto al consumo nel suo complesso. L'approccio **"top-down"** va dal generale allo specifico, quindi utilizzano la stima dell'energia totale (es. di un Computer) e altre variabili pertinenti per attribuire il consumo di energia a delle caratteristiche specifiche (es. CPU). L'approccio **"bottom-up"** comincia dallo specifico per trarre conclusioni generali. Questo tipo di approccio calcola il consumo energetico di singole caratteristiche (es. server, rete, storage, e infrastruttura) per poi estrapolare dei risultati che rappresentano l'intero sistema (es consumo di un intero Data Center).

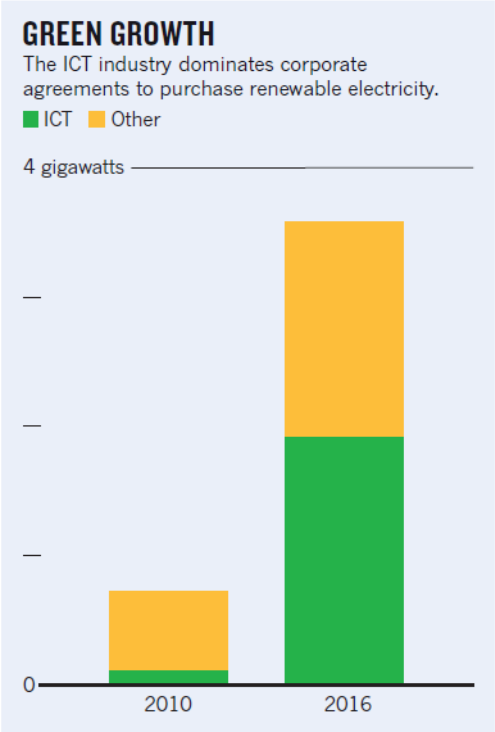


Figura 3.14: Trend utilizzo energia rinnovabile (2010 - 2016) [34].

Per le motivazioni elencate nelle sezioni precedenti 3.2.1, molti studi sono stati fatti per la modellazione e la predizione dei consumi di un Data Center.
Nella figura 3.15 è schematizzata una classificazione delle varie ricerche presenti in letteratura, suddivisi a seconda delle proprietà prese in considerazione per gli studi. Una prima suddivisione generica è Hardware e Software, per poi andare sempre più nel dettaglio arrivando ai singoli componenti di un server (es. processore).

In particolare possiamo distinguere questi tipi di modelli [28]:

- **Vista aggregata dei modelli di consumo dei server**
  - `Modelli Additivi`
    Il modo più diretto per modellare il consumo energetico è sommare i vari contributi in termini di potenza di tutti i componenti [36]:

$$P_{totale} = P_{CPU} + P_{memoria} + P_{disco} + ... + P_{I/O}$$

    dove P_{CPU}, $P_{memoria}$, $P_{I/O}$ e P_{disco} rappresentano rispettivamente il consumo del processore, memoria, "Input/Output" e lo "storage". Inoltre, ogni

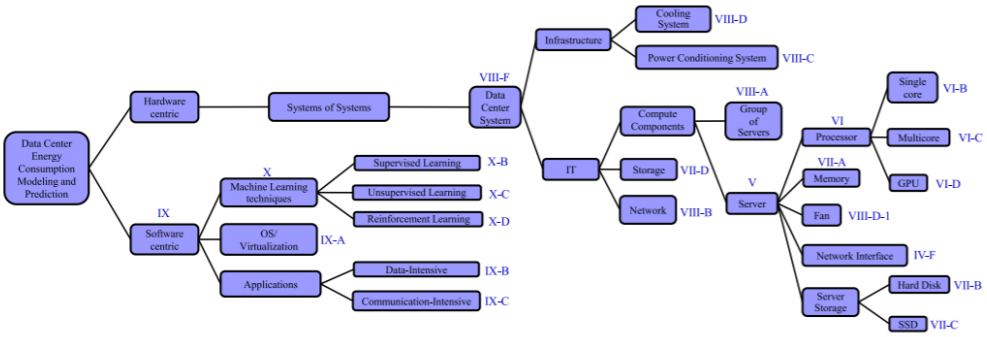


Figura 3.15: Tipi di modellazione del consumo energetico di un data center [28].

componente ha un proprio modello di consumo energetico. Ad esempio, il consumo di energia della CPU è una funzione della tensione, frequenza e utilizzo; il consumo di energia della memoria è una funzione della sua dimensione, frequenza, e l'utilizzo, il consumo di energia del disco rigido è una funzione della velocità di lettura e velocità di scrittura.

- `Modelli basati sul workload del sistema`
  Questi modelli calcolano il consumo di un server considerando il workload delle risorse che lo compongono. Tradizionalmente la CPU è il più grande consumatore di energia in un server, quindi la maggior parte dei modelli utilizzano come metrica il "workload" del processore per modellare il consumo dell'intero sistema.

- `Altri Modelli`
  I modelli additivi e quelli basati sull'utilizzo rappresentano la maggior parte dei modelli che stimano il consumo di potenza del server. Tuttavia, ci sono più altri modelli di potenza che non possono essere specificatamente attribuiti a queste due categorie.

- **Modelli di consumo del Processore**
  Oggi, la CPU è uno dei maggiori consumatori di energia di un server. I processori moderni come lo Xeon Phi consistono in miliardi di transistor che utilizzano un enorme quantità di energia. È stato dimostrato che il consumo di un server può essere descritto da una relazione lineare tra il consumo energetico e l'utilizzo della CPU. La frequenza della CPU decide in larga misura la potenza utilizzata da un processore.
  Tra questi modelli abbiamo:

– `Approccio alla modellazione del consumo di un processore`
Il consumo energetico di un processore può essere modellato come consumo statico e dinamico. E' stato osservato che i processori che presentano una bassa attività presentano una potenza statica molto grande rispetto a quella dinamica. Questo a causa delle perdite nelle tecnologie submicroniche. Nella figura 3.16 c'è un esempio reale di questo fenomeno. Si può chiaramente osservare che il processore multicore ha una quantità significativa di dispersione di potenza. Si può inoltre notare che le cache del processore contribuiscono significativamente al consumo energetico dello stesso.

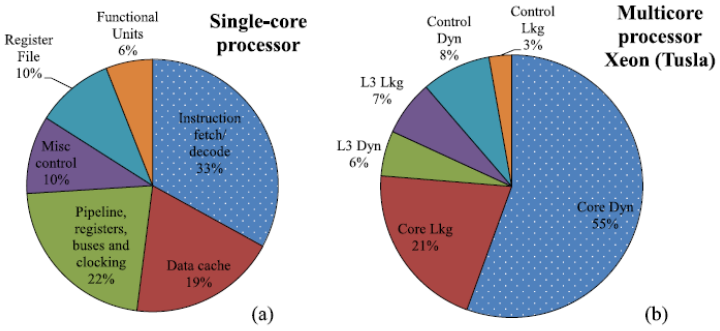


Figura 3.16: Ripartizione del consumo di energia di un processore single-core (a) e multicore (b) [28]

Esistono due approcci di alto livello per la generazione di modelli di potenza per i processori: (1) `modelli a livello di circuito` (2) `modellazione statistica`.
La modellazione a livello di circuito è più accurata ma ha un approccio computazionalmente molto costoso. La modellazione statistica ha un alto costo iniziale mentre il modello viene addestrato, ma è molto più veloce da usare. Approcci statistici per la modellazione della potenza del processore si basano sull'analisi dati condotta sui contatori delle prestazioni del processore.

– `Modelli di consumo di CPU Single-Core`
Viviamo in un'era di processori multicore, ma esistono molti modelli di potenza single-core. Questo perchè molti degli attuali modelli di potenza del processore multicore sono reincarnazioni di modelli di potenza single-core. Esistono modelli additivi e quelli che utilizzano i contatori di prestazioni.

  - `Modelli di consumo di CPU Multicore`
    La maggior parte degli attuali server sono dotati di CPU multicore.

    Visto che l'uso di core diversi da thread diversi possono creare vari livelli di consumo di risorse, è importante modellare il consumo energetico a livello di core della CPU. Il consumo energetico del server dipende dalla velocità con cui lavorano i core. Attualmente ci sono due principali approcci per la modellazione del consumo energetico di un processore multicore: (1) `teoria delle code` (2) `ripartizione "saggia" del consumo dei componenti`.
  - `Modelli di consumo della GPU`
    Le "Graphical Processing Units" stanno diventando dei componenti molti diffusi nei data center. Questo perché molti server moderni sono dotati di "General Purpose Graphical Processing Units" (GPGPU) per consentire l'esecuzione di enormi quantità di sistemi concorrenti supportati da hardware. Nonostante l'importanza delle GPU, la misurazione, la modellazione e i modelli di predizione del loro consumo energetico sono ancora agli inizi.

    Dalla Figura 3.17 si può osservare che i core della GPU (unità funzionali) dominano il consumo di energia e che la memoria "off-chip" consuma una notevole porzione di potenza in una GPU.

- **Modelli di consumo di memoria e storage** Tradizionalmente, i processori sono sempre stati considerati i principali contributori al consumo energetico del server. Tuttavia, negli ultimi anni il contributo apportato dalla memoria (RAM) e dalla memoria secondaria (HDD / SSD / storage dei server) per il consumo energetico del data center è aumentato considerevolmente. Di seguito i modelli:

  - `Modelli di consumo della memoria`
    La memoria è il secondo più grande consumatore di energia in un server. In alcuni sistemi arriva ad essere ad essere il più energivoro (Figura 3.18).
  - `Modelli di consumo di HDD`
    "Hard Disk Drive" è attualmente il tipo principale di unità di supporto secondaria di memorizzazione utilizzata nei server dei data center. Il disco è il sottosistema più difficile da modellare correttamente per la mancanza di visibilità sugli stati di alimentazione del disco stesso e per l'impatto delle cache sui consumi. Ci sono tre fonti di consumo energetico all'interno di un HDD: (1) *"Spindle Motor" (SPM)* (2) *Voice Coil Motor (VCM)* (3) *l'elettronica*.
  - `Modelli di consumo di SSD`
    I dischi a stato solido (noti anche come "Solid State Drive") stanno

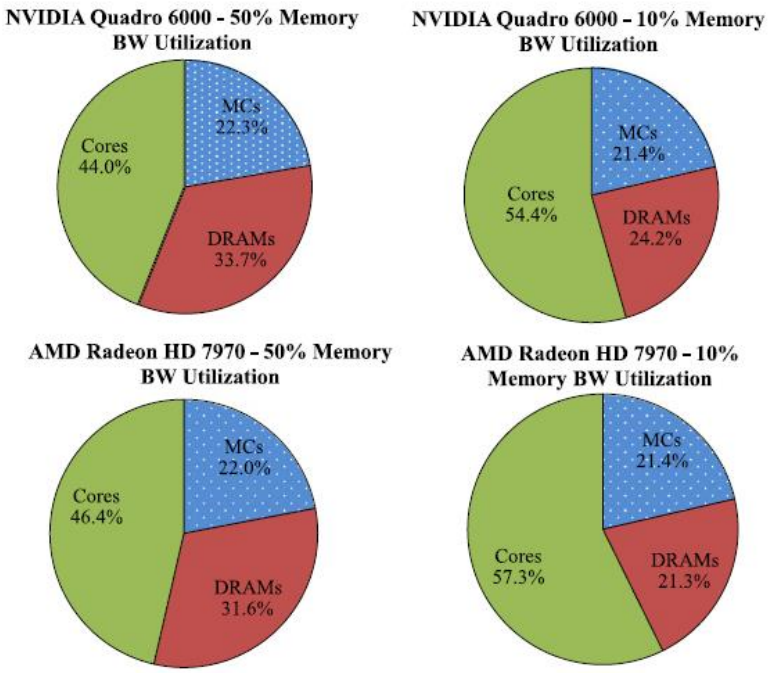


Figura 3.17: Ripartizione del consumo di energia di GPU NVIDIA e AMD [28]

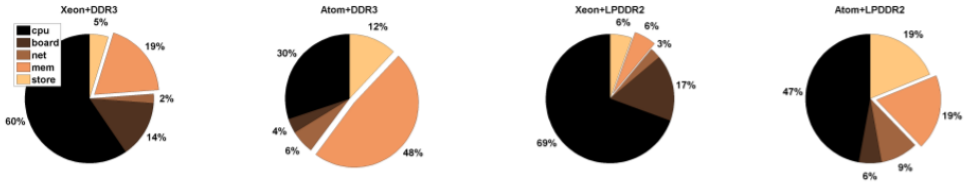


Figura 3.18: Ripartizione dei consumi nei componenti di un server. [37] La RAM, in alcune configurazione (es Atom) arriva ad essere il componente più energivoro.

diventando una componente sempre più diffuso nei data center più moderni (Figura 3.19 (a)). SSD è diventato un forte candidato per l'archiviazione primaria grazie alla sua migliore efficienza energetica e all'accesso casuale più rapido (Figura 3.19 (b)).

– `Modelli di consumo dello storage di server`
I sistemi di storage implementati nei data center rappresentano una notevole quantità di energia consumata da un intero centro dati. Alcuni

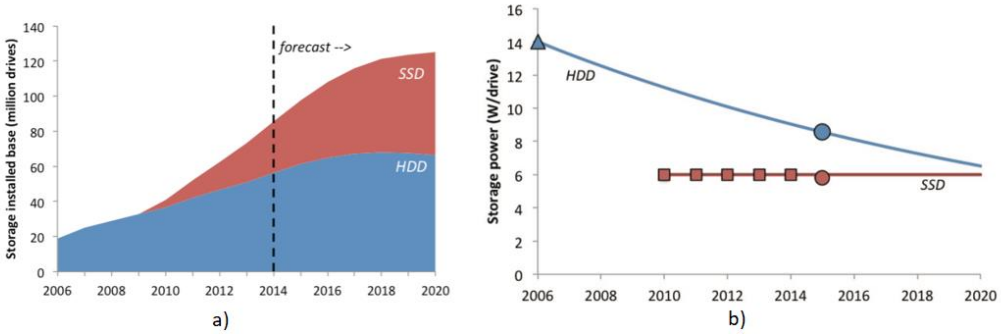


Figura 3.19: Nella a) la diffusione del SSD che sta pian piano sostituendo l'utilizzo dei tradizionali HDD; nella b) un confronto di consumo tra HDD e SSD [38]

studi hanno dimostrato che l'energia consumata dallo storage può arrivare fino al 40% dell'energia consumata da un intero DC. Lo storage di un data center è costituita da un controller di archiviazione e da uno storage "directly-attached". Il consumo energetico dei sistemi di storage è unico perché contengono grandi quantità di dati (spesso conservano diverse copie dei dati in livelli di storage superiori). Nei sistemi di archiviazione di backup, la maggior parte dei dati è inutilizzata perché i backup sono generalmente accessibili solo quando si verifica un errore nel livello di archiviazione superiore.

- **Modelli di consumo di un Data Center**
  Un Data Center è composto da più elementi come gruppi di server, storage, dispositivi di rete, unità di distribuzione dell'alimentazione e sistemi di raffreddamento. Ognuno di questi ha un consumo intrinseco che viene studiato e modellato.

  - `Modelli di consumo di un gruppo di Server`
    I modelli di potenza per un gruppo di server possono essere classificati in tre sottocategorie: (1) `modelli di potenza basati sulla teoria delle code` (2) `modelli di potenza basati su metriche di efficienza` (3) `altri`
  - `Modelli di consumo della rete di un DC`
    I modelli di consumo di una rete tengono in considerazione la rete LAN che collega un gruppo di server all'interno di uno stesso Data Center, e anche il "network" che collega diversi DC distribuiti su reti geografiche.

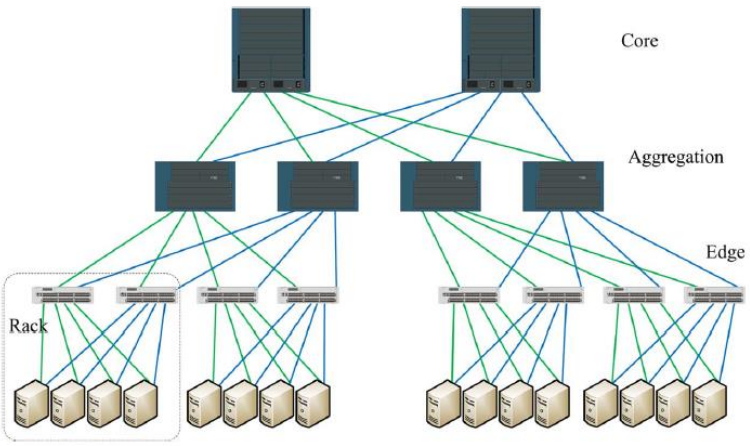


Figura 3.20: Una tipica rete di un Data Center [28]

Come osservabile dalla figura 3.20 a concorrere al consumo di una rete abbiamo due principali attori: (1) **consumo dei link** (2) **consumo dei device di rete**

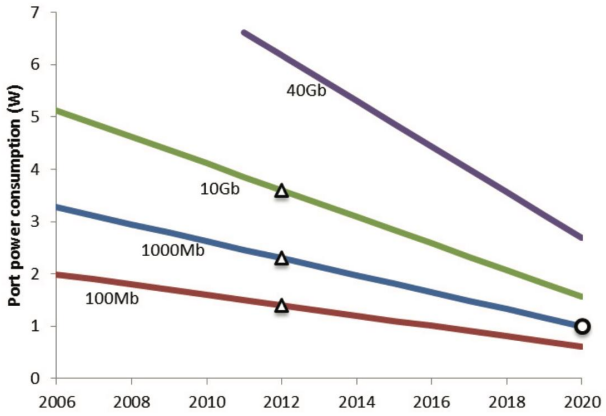


Figura 3.21: Potenza consumata da porte con diversa velocità [38]

I dispositivi di rete utilizzano energia in relazione al numero di porte e alla loro velocità 3.21.

La rete di un DC è la struttura con la quale i processori, le memorie e i dispositivi di I/O sono condivisi dinamicamente ed è generalmente considerato un elemento di progettazione critico dell' architettura. In questa struttura i dispositivi di rete come router, switch, ecc. svolgono

un ruolo fondamentale nelle varie operazioni di un data center. Nella maggior parte delle situazioni i dispositivi di rete come i router sono predisposti per i carichi di punta, ma mediamente operano a bassi livelli di utilizzo. Il router potrebbe arrivare a consumare tra 80-90% della sua potenza di picco quando non inoltra alcun pacchetto. Pertanto, è necessario adottare misure per modellare il consumo energetico di questi dispositivi per consentire quindi lo sviluppo di tecniche operative efficienti dal punto di vista energetico.

– `Modelli di consumo dei sistemi di alimentazione`
Il sistema di alimentazione è responsabile della fornitura di energia elettrica ai dispositivi di un data center. Il sistema di alimentazione di un data center consuma notevoli quantità di energia, così anche la potenza sprecata durante il processo di trasformazione. La distribuzione di energia elettrica ininterrotta in un data center richiede parecchie infrastrutture (quali trasformatori, quadri, PDU, UPS, ecc.). In una tipico DC, un quadro elettrico primario distribuisce potenza tra più sottostazioni di "Uninterrupted Power Supply" (UPS). Ogni UPS a sua volta, fornisce alimentazione ad un insieme di PDU. Un PDU è associato ad un insieme di server rack e ciascun rack dispone di diversi chassis che ospitano i singoli server. Un'illustrazione di tale gerarchia è mostrata in Figura 3.22

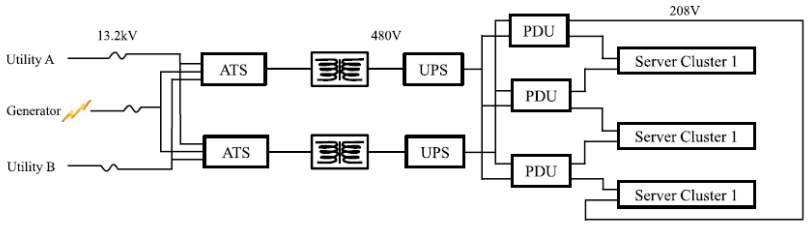


Figura 3.22: Un esempio di un sistema di alimentazione di un data center ad alta disponibilità con percorsi di distribuzione ridondante. [28]

Le PDU sono responsabili della fornitura di un'alimentazione adeguata per i server; trasformano la potenza ad alta tensione distribuita in tutto il data center, a livelli di tensione appropriati per i server. Le PDU subiscono una perdita di potenza costante che è proporzionale al quadrato del carico. Gli UPS invece fungono da utenze elettriche temporanee durante le interruzioni di corrente.

– `Modelli di consumo dei sistemi di raffreddamento`

Anche un sistema in rack accuratamente progettato che utilizza componenti a bassa tensione arriva a consumare una potenza di circa 22 kW. Questi consumi generano un notevole calore che deve essere smaltito affinché i server funzionino in un intervallo di temperatura di esercizio sicuro. Per questo vengono utilizzati sistemi di raffreddamento che mantengono efficacemente la temperatura di un data center. La potenza di raffreddamento è il più grande consumatore del "non computing" in un data center seguita dalla conversione dell'alimentazione e da altre perdite. La figura 3.23 fornisce una ripartizione del consumo del sistema di condizionamento in un DC. La potenza di raffreddamento dipende da molti fattori come il layout del data center, la portata d'aria ecc.

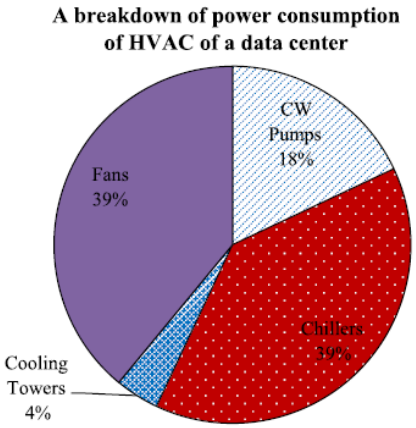


Figura 3.23: Ripartizione del consumo di energia di un sistema di HVAC di un DC. I tre principali consumatori di energia includono ventilatori (39%), refrigeratori (39%) e pompe di raffreddamento dell'acqua (18%). [28]

- **Modelli di consumo Software**
  Oltre ai modelli che stimano il consumo in base alle caratteristiche fisiche dei data center, esistono dei modelli che considerano il tipo di applicazioni e il workload. Il software di un data center può essere classificato in cinque categorie: (1) `ad alta intensità di calcolo` (2) `ad alta intensità di dati` (3) `ad alta intensità di comunicazione` (4) `sistema operativo e software di virtualizzazione` (5) `software generico`. Nei task "computation intensive" il principale consumatore di risorse sono i core della CPU. In attività "data-intensive" le risorse di archiviazione del sistema cloud sono i principali consumatori di energia. In task "communication-intensive" una

gran parte dell'energia viene utilizzata dalle risorse di rete come schede di rete, router, switch ecc..

– `Modelli di consumo a livello di OS e Virtualizzazione`
  Un sistema operativo (OS) si trova tra i due layer chiave dello stack di un data center: l'hardware fisico e le applicazioni. Le applicazioni creano la domanda di risorse ed è invece l'hardware fisico che effettivamente consuma la potenza informatica. Il sistema operativo è considerato l'intermediario tra i due strati chiave. Inoltre con la diffusione della virtualizzazione si è introdotto un ulteriore Layer tra l'harware e le app: l'**"hypervisor"** (Figura 3.24).

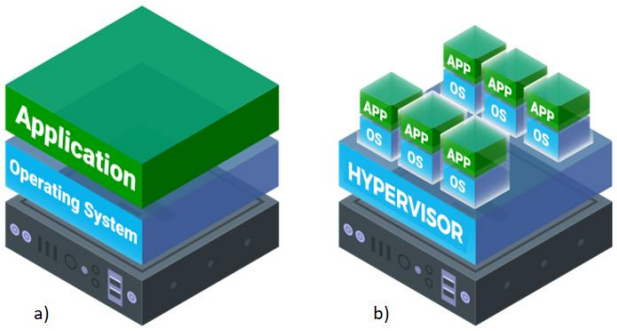


Figura 3.24: Vari layer di un architettura tradizionale (a) e virtualizzata (b)[28]

Alcune ricerche hanno studiato i consumi di diversi hypervisor secondo tre dimensioni ortogonali: hardware (ARM64, X86_64), hypervisor (Microsoft Hyper-V, VMware ESXi, KVM/QEMU e XenServer, nonchè il container engine Docker) e workload ("memory intensive", "computing intensive" e workload misto) [39]. Gli hypervisor mostrano consumi energetici diversi sullo stesso hardware che eseguono lo stesso carico di lavoro. Sebbene gli hypervisor tradizionali abbiano diverse efficienze energetiche, nessun singolo hypervisor supera gli altri hypervisor su tutte le piattaforme in termini di consumo di energia. Nella figura 3.25 vediamo un confronto tra consumo di server fisici e virtuali; utilizzando tecnologie di virtualizzazione si risparmia più della metà rispetto alla potenza richiesta da server fisici [32].

Alcuni studi si spingono all'analisi di consumo della *containerizzazione*. Sebbene la virtualizzazione dei container sia considerata una virtualizzazione "light" in termini di implementazione e manutenzione, essenzialmente non è più efficiente dal punto di vista energetico rispetto alla tecnologia di virtualizzazione convenzionale.

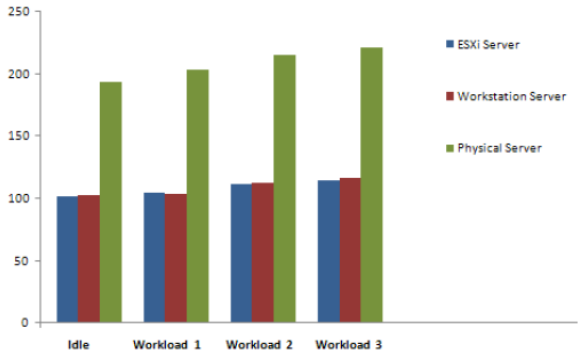


Figura 3.25: Differenza nel consumo energetico tra server fisici e virtualizzati [32]

- `Modelli di consumo di applicazioni "data intensive"`
  I task "data intensive" sono generalmente legati all' I/O e richiedono elaborazioni di grandi volumi di dati. Possono essere classificati come *online* e *offline* in base al loro tipo di operazioni. La maggior parte dei modelli possono essere classificati sotto il secondo tipo, con molti modelli di potenza riferiti a MapReduce. Un "datawarehouse" invece è un esempio di applicazione "data-intensive" offline che viene distribuita frequentemente nei cluster di un data center.

- `Modelli di consumo di applicazioni "communication intensive"`
  Le applicazioni "communication intensive" sono composte da un gruppo di task che scambiano una grande quantità di messaggi tra di loro. Queste applicazioni impongono un onere significativo all' infrastruttura di rete di un data center. .

- `Modelli di consumo di generali applicazioni`
  In questa categoria rientrano i modelli di applicazioni che non appartengono alla classificazione descritta in precedenza.

3.2.2 Server

I server sono gli elementi di un data center che, dopo i sistemi di raffreddamento, consumano più energia (Figura 3.10). Per questo motivo in letteratura troviamo molti articoli riguardanti i server, metodi per il risparmio energetico e modelli per la predizione dei consumi.

CPU

In un server, il processore è il più grande consumatore di energia con un consumo di circa il 32% della potenza totale. E' seguito dalle periferiche 20%, l' alimentazione 15%, la memoria 14%, ecc..
Negli anni la *legge di Moore* ha permesso all'industria ICT di aumentare notevolmente le prestazioni, la velocità dei chip e ridurre la potenza consumata dei processori. Questi miglioramenti hanno mantenuto l'esponenziale crescita della domanda in parte sotto controllo in termini di consumi energetici.

Il consumo energetico del processore dipende dalla sua tensione, frequenza e utilizzo [40].

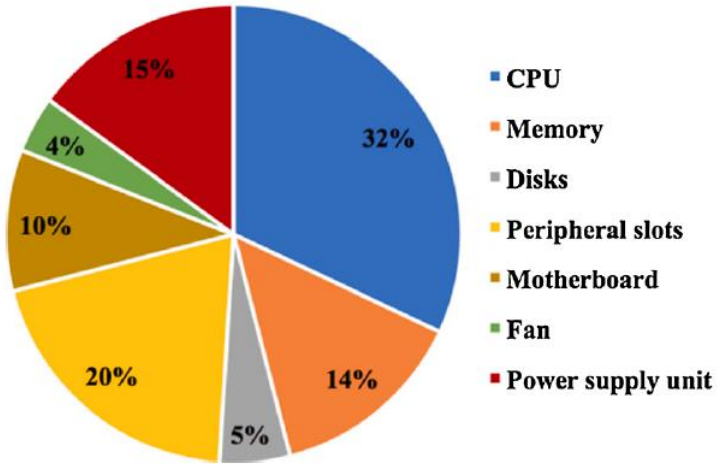


Figura 3.26: Composizione del consumo energetico di un server [40]

La Figura 3.27 mostra il consumo energetico dei principali sottosistemi per un server di Google al variare del carico di lavoro che varia dal livello di inattività ("idle mode") a quello di piena attività (CPU al 100%). A qualsiasi workload la CPU, utilizza due terzi dell'energia al picco di utilizzo e circa il 40% in "idle"; è sempre il maggior consumatore di energia.

Per i server, la CPU non è solo il più grande consumatore di potenza, ma è anche il responsabile della maggior parte delle modifiche delle dinamiche al consumo energetico del sistema [41]. Il consumo di un server dipende dal tipo di processore e soprattutto dal suo carico di lavoro. Come è possibile osservare nella Figura 3.30, per esempio, i sistemi con processori Intel Xeon Platinum 8180 e 8280 consumano una maggior quantità di energia rispetto agli altri.
Il consumo cresce in modo quasi proporzionale con il workload. Questo aspetto di

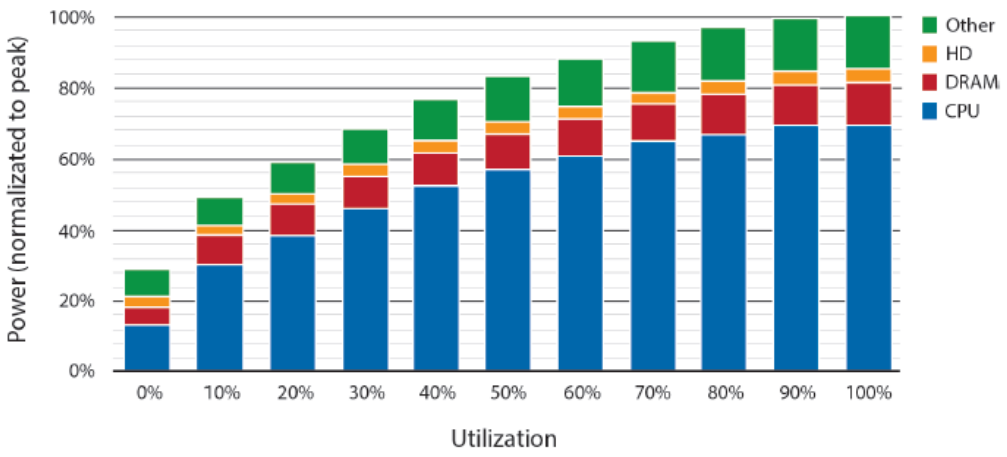


Figura 3.27: Consumo energetico dei sottosistemi di un server ×86 con workload variabile (da "idle" a pieno carico) [31]

proporzionalità è stato introdotto da Barroso e Hölzle [31]. Idealmente, i sistemi "energy-proportional" consumano poca energia quando sono inattivi (in particolare in stati di inattività dove restano disponibili a lavorare) e consumano gradualmente più energia all'aumentare del carico di lavoro. La curva ideale che rappresenta questa proporzionalità è assumere la linearità tra workload e consumo di energia. L'intuizione chiave del modello proporzionale è stato che avvicinando l'efficienza del server al massimo teorico ad ogni workload, sarebbe migliorata l'efficienza complessiva. Assumendo che il consumo energetico del server sia ridimensionato proporzionalmente al carico di lavoro, l'efficienza è ottimizzata su una gamma più ampia di utilizzo, come mostrato nella Figura 3.28.

La figura a sinistra mostra il consumo energetico di un server (ca. 2006) la cui potenza in idle è del 70% della potenza di picco. Poiché il consumo energetico non si adatta al carico di lavoro, l'efficienza è molto inferiore nella maggior parte delle condizioni operative. La figura a destra mostra un server con una potenza "idle" che è del 20% del picco. In questo caso l'efficienza è molto più elevata in tutti i punti di utilizzo. Nel 2007 la maggior parte dei server consumava quasi la stessa potenza con un utilizzo dello 0% (ovvero, senza fare nulla) e con un utilizzo del 100% (cioè a massimo carico). Negli anni la proporzionalità dei sistemi è aumentata, migliorando di fatto l'efficienza rispetto a quella dei predecessori (Figura 3.29).
Fino ad ora abbiamo parlato del criterio di proporzionalità riferendoci al solo processore, ma per migliorare i consumi è necessario che tutti i componenti hardware di un sistema (es. RAM, storage e networking) rispettino questo aspetto.
Anche il software svolge un ruolo importante per l'efficientamento energetico:

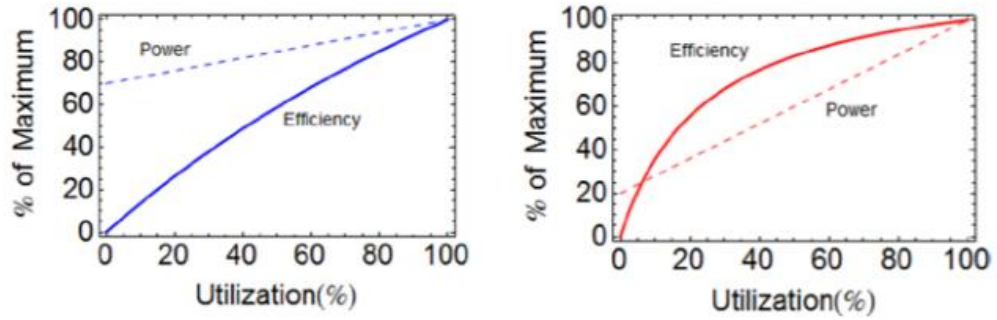


Figura 3.28: Il consumo energetico e l'efficienza di due modelli di server [25]

sistemi di "power managment" e "scheduling" più intelligenti hanno un ruolo molto importante.

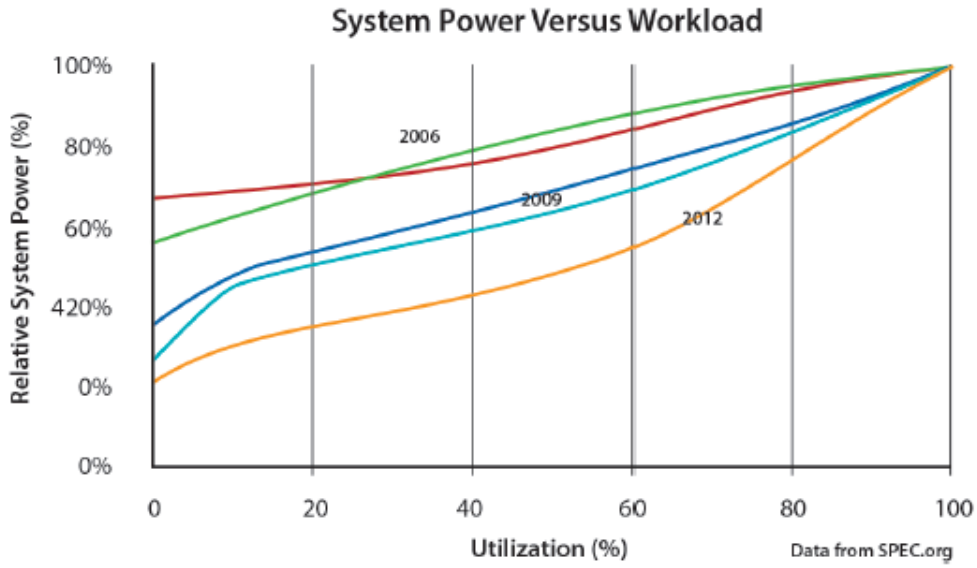
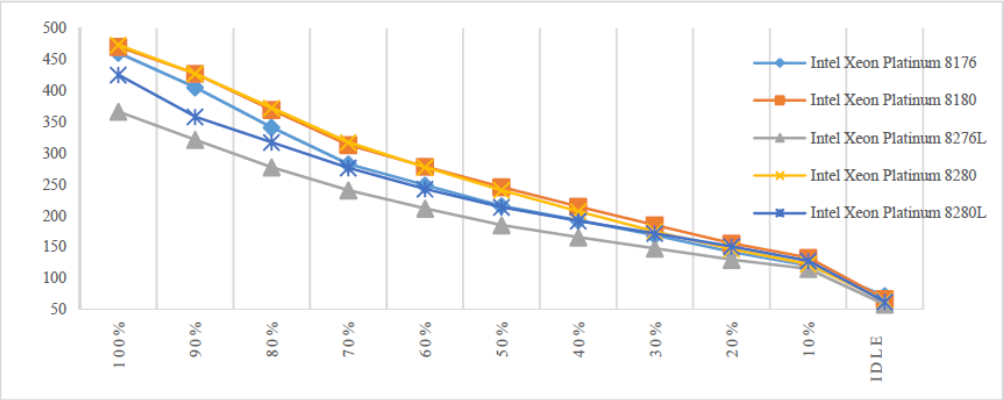


Figura 3.29: Confronto negli anni della proporzionalità dei sistemi (2006 - 2012) [25]

Sulla proporzionalità del processore sono stati fatti molti passi avanti, infatti come vediamo nella figura **??** la relazione consumo/workload CPU può essere



Consumo energetico di alcuni processori Intel Xeon Platinum sottoposti a diversi workload

Figura 3.30: [42]

approssimato da una relazione lineare [43]

$$P_{tot} = P_{idle} + (P_{busy} - P_{idle})u$$

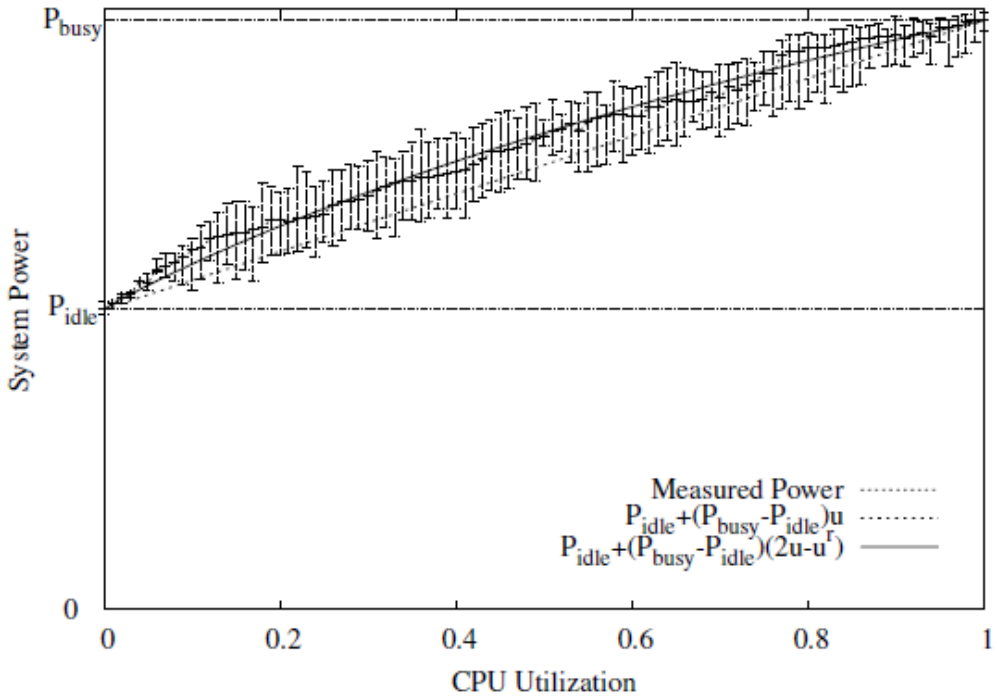
dove P_{tot} è la potenza consumata dal server, P_{idle} è la potenza consumata quando il server è in stato di "idle" e P_{busy} è la potenza consumata a pieno carico.

3.2.3 Desktop

Fino ad ora ci siamo concentrati sui Data Center e i Server che mettono a disposizione vari servizi. A richiedere tali servizi, c'è un numero sempre crescente di device personali (quali smartphone, desktop, laptop ecc.) e dispositivi IoT.

Trend

Nel 2019, si stima che quasi la metà delle famiglie nel mondo avesse un computer in casa. Nei paesi in via di sviluppo, il tasso di penetrazione dei PC è inferiore con circa un terzo delle famiglie che possiedono un computer. Al contrario, la quota di famiglie con un personal computer nei paesi sviluppati è dell'80%. In generale, la quota di famiglie con un computer è aumentata costantemente in tutto il mondo poiché l'uso del computer e l'accesso a Internet stanno diventando più diffusi [44]. Questo trend è stato ulteriormente accentuato durante il lockdown Covid-19, iniziato nel febbraio 2019. Con la diffusione della pandemia, quasi tutti gli stati hanno



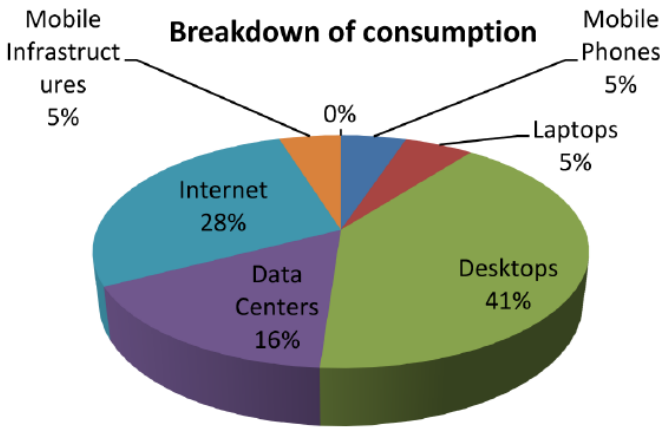
Relazione workload CPU / consumo [43]

implementato un lockdown, chiudendo le attività che richiedono interazioni tra persone - incluse università, scuole, centri commerciali, templi, uffici, aeroporti e stazioni ferroviarie. Il lockdown ha portato la maggior parte delle persone ad utilizzare Internet e i servizi online per comunicare, interagire e continuare a lavorative da casa. I servizi Internet hanno visto un aumento dell'utilizzo dal 40% al 100%, rispetto ai livelli precedenti al lockdown. I servizi di videoconferenza come Zoom hanno visto un aumento di dieci volte il loro normale utilizzo; meeting online per i lavoratori e lezioni da casa per gli studenti sono stati i maggiori complici di questo fenomeno. I blocchi in tutti i paesi hanno comportato un aumento dell'uso dei sistemi informativi con enormi cambiamenti nell'utilizzo [45].

Consumi

Tradizionalmente, i data center e le infrastrutture mobile sono state considerate le maggiori cause di consumo all'interno dell'area ICT e la maggior parte degli efficientamenti energetici sono stati specificamente mirati su queste aree. Tuttavia,

i risultati nella figura 3.31 mostrano che i dispositivi personali consumano energia paragonabile se non di più a quella dei DC [46].



Ripartizione dei consumi per tipi di device. I device personali hanno un consumo predominante rispetto a quello dei DC. Nei desktop circa la metà dei consumi è dovuta alla presenza dei monitor.

Figura 3.31: [46]

Mentre la policy europea presuppone che l'efficienza continui senza sosta almeno fino al 2050, alcuni esperti avvertono che i miglioramenti dell'efficienza nelle tecnologie dei processori stia raggiungendo un limite (*legge di Moore*) e potrebbe rallentare dopo il 2025; abbiamo raggiunto un livello in cui la miniaturizzazione dei transistor si scontra con i limiti fisici. Questo perché transistor più piccoli hanno strati isolanti più sottili che portano a una perdita di corrente elettrica e ciò richiede più potenza per compensare la corrente persa (quindi più consumo). Se i miglioramenti dell' efficienza fino ad oggi hanno compensato l'impatto dovuto dall'aumento della domanda, un limite a ulteriori efficienze minaccia una futura impennata delle emissioni di CO_2 [47].

E' quindi utile ricercare dei modi per ottenere un funzionamento efficiente dal punto di vista energetico anche per questi device.

Sono stati condotti molti studi che analizzano il consumo dei computer.

La Figura 3.33 illustra il consumo di potenza durante un tipico giorno della settimana e un weekend per due computer desktop con capacità computazionale ed età molto diverse.

Desktop 1 è un computer di 3 anni con caratteristiche base (Processore 2.3 GHz Intel Core Duo), generalmente utilizzato per l'esecuzione programmi come elaboratori di testi e fogli di calcolo. Il *Desktop 2* è invece un computer ad alte

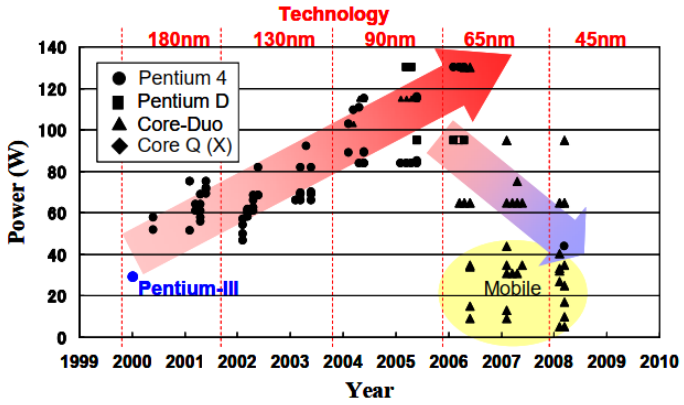


Figura 3.32: Consumo e tecnologie CPU [48]

prestazioni con un processore multi-core (Processore 3.4 GHz Intel Xeon) utilizzato per eseguire modellazione 3D e programmi complessi che richiedono un'elevata potenza di elaborazione.

Come si nota dalla figura 3.33, *Desktop 1* consuma molta meno energia rispetto a *Desktop 2*. *Desktop 1* ha un consumo molto stabile durante il giorno, a differenza di *Desktop 2* la cui potenza è oscilla in diversi momenti della giornata.

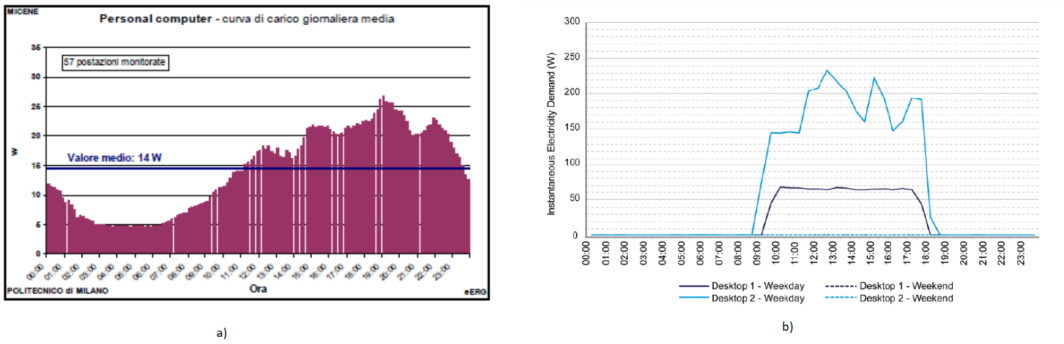


Figura 3.33: (a) Consumi di due desktop monitorati in ufficio [49]; (b) Curva di carico di Personal Computer in ambito residenziale [50]

Diverso è l'utilizzo domestico misurata dal progetto Micene [51] dove vediamo un utilizzo concentrato nelle fasce serali (Figura 3.33). L' uso dei Computer Desktop è in media un paio di ore al giorno; le altre ore risulta in stand by (più del 20% del consumo mensile deriva da quest' ultimo stato) [52]. Notiamo quindi una

distinzione tra computer utilizzati in un ambiente d'ufficio e computer utilizzati in ambiente domestico, con modelli di utilizzo in questi contesti differenti.

OS e Software

La crescente preoccupazione per l'aumento dei consumi dell' ICT indica la necessità di modellizzazione e stima della potenza per tutti i componenti di un sistema. Un componente è quindi il **Software**, che si presenta sia come `sistema operativo (OS)` che come `applicazioni utente`. L'esecuzione del Software guida le attività dell'hardware sottostante e, il modo con cui il software utilizza l'hardware può avere un impatto notevole sulla dissipazione di potenza di un sistema [53].

Se consideriamo un Desktop, l'hardware e il sistema operativo hanno un importante ruolo nella gestione globale dell'energia, ma non sono gli unici componenti a dettare il consumo. Su questi computer è possibile installare una moltitudine di programmi che avranno un impatto sulla gestione dell'energia. Ad esempio, se un software di terze parti utilizza una determinata periferica in modo errato, potrebbe aumentare la sua domanda di energia anche quando non è necessaria. L'idea di fondo è che il consumo di energia dipende non solo dalle caratteristiche hardware, ma anche dall'utilizzo del software e dalle caratteristiche interne del software. Ad esempio, un software più complesso richiederà più cicli di CPU, oppure una singola operazione di scrittura lunga su disco può consumare meno energia rispetto a diverse piccole operazioni di scrittura [54].

Uno studio condotto dal Politecnico di Torino, ha analizzato la variazione del consumo in diversi scenari elencati nella Tabella 3.2.3:

	Scenario	Categoria
0	Idle	Idle
1	Navigazione Web	Rete
2	Email	Rete
3	Suite Office	Produttività
4	Trasferimento Dati (disco)	File System
5	Trasferimento Dati (USB)	File System
6	Presentazione	Multimedia
7	Chiamata Skype (no video)	Rete
8	Chiamata Skype (video)	Rete
9	Multimedia (Audio)	Multimedia
10	Multimedia (Video)	Multimedia
11	Peer-to-Peer	Rete

Dai risultati dell'analisi visualizzati in figura 3.34, si può vedere che i Desktop più moderni pur essendo più efficienti dal punto di vista energetico negli stati di standby

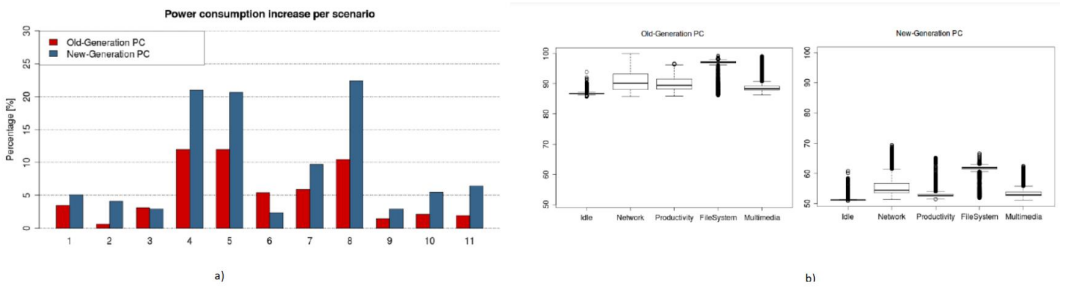


Figura 3.34: (a) Aumento del consumo energetico per scenario rispetto allo stato "Idle" (in percentuale) [54]; (b) Box Plot per ogni categoria di scenario [54]

e di inattività, grazie alla loro maggiore scalabilità, sono più sensibili all'impatto dell'utilizzo del software sul consumo energetico. Ciò indica che la ricerca dovrebbe concentrarsi sulla riduzione di questo impatto, poiché sarà sempre più significativo con il passare del tempo. Si può inoltre notare che scenari appartenenti alla stessa categoria (es Rete) hanno consumi molto diversi tra loro quindi non è possibile fare delle considerazioni generali sul consumo di software appartenenti alla stessa categoria.

3.2.4 Laptop

Molte delle funzioni di efficienza energetica dei computer desktop sono state originariamente sviluppate per i portatili per prolungare il tempo di durata della batteria. I computer portatili consumano una frazione dell'energia dei computer desktop. Questo è dovuto a una serie di fattori [55]:

- Assorbono meno potenza;
- Vanno in modalità di risparmio energetico più rapidamente rispetto al computer desktop per preservare la carica della batteria;
- Sono più frequentemente spenti e scollegati rispetto ai computer desktop.

Nella figura 3.35 ci sono le misurazioni di tre laptop con età e caratteristiche diverse. *Laptop 1* di tre anni con un processore singolo Intel Centrino da 1.3 GHz, *Laptop 2* di due anni con 2.3 GHz Intel Core Duo e *Laptop 3* con processore Intel Core i5 da 2.6 GHz. Come visto, il laptop più recente (*Laptop 3*) ha il fabbisogno energetico complessivo più basso, nonostante i suoi picchi occasionali durante il giorno. Anche in stand-by i laptop più moderni consumano meno di quelli più vecchi.

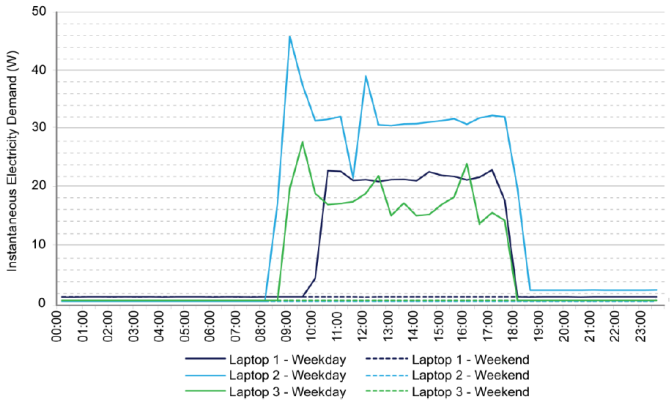


Figura 3.35: Misurazione di consumo giornalieri di computer laptop in ambiente d'ufficio [49]

Il consumo dei Laptop è fortemente influenzato dalla grandezza del monitor [48] e dalla sua luminosità [56].

Software e OS

Come per i Desktop, anche per i Laptop il Sistema Operativo e le applicazioni hanno un ruolo chiave nei consumi di un sistema. Windows ha un consumo inferiore sia in modalità On che "Idle" (Figura 3.36) rispetto a Linux.

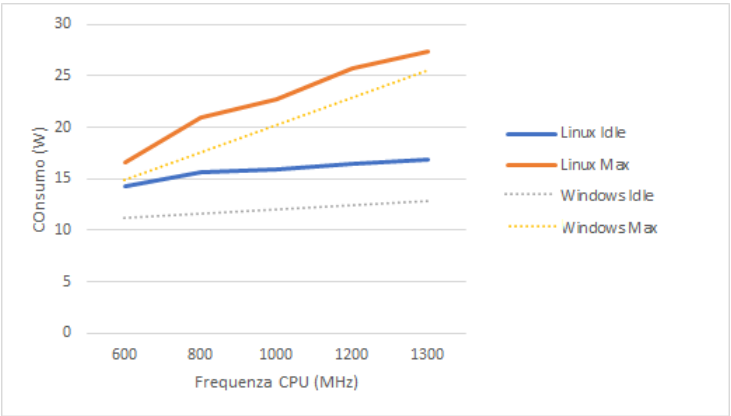


Figura 3.36: Sistemi Operativi a confronto [56]

Nel 2011 è apparso un post su MSDN Blog [1] che riguardava la misurazione del consumo energetico di vari browser Internet. Gli autori hanno misurato il consumo energetico e la durata della batteria di un comune laptop in sei scenari diversi, per esempio navigazione su about:blank (consumo energetico dell'interfaccia utente del browser),caricamento di siti Web di notizie popolari (scenario HTML4 comune), esecuzione di HTML5 Galactic esperienza (rappresentante dello scenario grafico HTML5)). La linea di base per il confronto degli scenari era Windows 7 senza alcun browser in esecuzione. Gli autori hanno eseguito le stesse operazioni con i diversi browser (IE9, Firefox, Opera e Safari), ottenendo risultati molto diversi sui consumi; quindi possiamo affermare che anche sui Laptop, il software può avere un impatto significativo sul consumo di energia.

Confronto di consumi tra Desktop e Laptop

Per i Desktop e i Laptop esistono varie modalità operative [57]:

1. `Modalità On`: normale funzionamento, il dispositivo è in esecuzione.

2. `Modalità standby`: il dispositivo è in grado di svegliarsi molto rapidamente; il consumo di energia è ridotto.

3. `Modalità ibernazione/sospensione`: modalità di sospensione profonda, il dispositivo è in sospensione; Consumo energetico notevolmente ridotto.

4. `Modalità off`: il dispositivo non svolge alcuna funzione; sembra spento ma consuma energia.

Come possibile vedere dalla figura 3.37 ogni modalità ha un consumo diverso; dalla modalità On che è la più energivora, a quella Off. In qualsiasi stato si trovino i PC, anche nella modalità Off, c'è sempre un consumo seppur molto ridotto.

La ripartizione di queste diverse modalità è molto diversa per Desktop e Laptop; Nella figura 3.38 si può vedere che la modalità predominante per i Desktop è quella On, per i Portatili quella "Low Power" quindi una modalità a basso consumo energetico.

Conclusioni

I consumi di Desktop e Laptop dipendono quindi da molti fattori:

[1] Browser Power Consumption - Leading the Industry with IE 9, http://blogs.msdn.com/b/ie/archive/2011/03/28/browser-power-consumption-leading-the-industry-withinternet- explorer-9.aspx

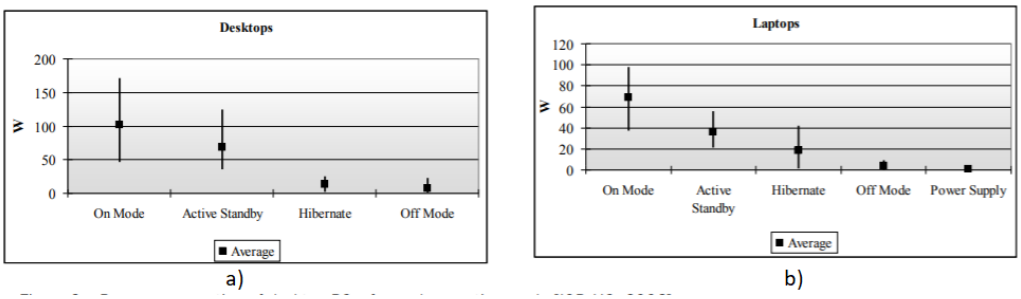


Figura 3.37: Consumo energetico per ogni modalità operativa dei PC desktop (a) e Laptop (b) [57]

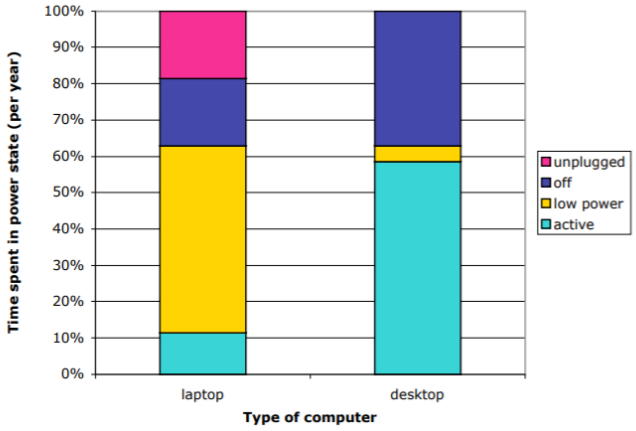


Figura 3.38: Tipico modello di utilizzo di computer portatili e desktop [55]

- `Differenti fabbisogni energetici dei singoli computer`
  Computer più moderni hanno un consumo inferiore rispetto a quelli più datati.

- `Diversi modelli di utilizzo e comportamento degli utenti`
  A parità di sistema e ore di utilizzo, il consumo dipende dal sistema operativo e dal software utilizzato. Inoltre bisogna tenere in considerazione il pattern di utilizzo; un PC d'ufficio sarà utilizzato presumibilmente 8 ore al giorno in orari lavorativi, un computer domestico invece sarà maggiormente utilizzato nelle fasce serali e nei week end.

Nel complesso le risorse, come la CPU, del computer risultano sempre sottoutilizzate [58].

Teniamo in considerazione che oltre ai consumi dovuti all'utilizzo dei PC c'è anche il costo energetico per produrlo; si stima che il consumo di produzione richiede 3 volte più energia rispetto all'utilizzo del PC per tre anni [48].

3.2.5 Altri dispositivi

Oltre ai Laptop e ai Desktop esistono molti altri dispositivi che consumano elettricità e potrebbero avere delle risorse da condividere. I più diffusi sono sicuramente tablet e smartphone, ma anche Raspberry, NAS, access point ecc. Alcuni di questi verranno trattati nella seguente sezione.

Smartphone e Tablet

Gli **smartphone** sono i device più utilizzati nell'arco di una giornata, nella vita odierna sono ritenuti indispensabili perchè combinano le funzionalità di un dispositivo di comunicazione tascabile con funzionalità simili a quelle di un PC. Infatti questi dispositivi vengono utilizzati per qualsiasi attività, dal comunicare con altre persone tramite chat, all' "home banking" allo shopping online, alla navigazione con browser, al controllo remoto dell'auto o di apparecchiature domestiche e anche per l'intrattenimento.

Si stima che la penetrazione degli smartphone (Android 72% e 26% iOS) nel 2020 fosse del 90% [44] con un trend in forte crescita mondiale [59]. Questi dispositivi sono connessi alla rete tramite WiFi o rete mobile. Si stima che in Europa circa l'87% delle case ha una connessione internet e che ci siano circa il 105% di abbonamenti mobile attivi [60].

Quasi l'85% della popolazione globale è coperta da una rete 4G, valore che raggiunge il 97% per l'Europa. La diffusione delle reti mobile (broadband e cellular) sta facendo calare l'utilizzo di quelle fisse; nella figura 3.39 vediamo il trend delle reti mobile e fisse.

Negli ultimi anni la natura della trasmissione dati sta cambiando rapidamente e si prevede che il traffico da dispositivi wireless e mobili rappresenterà oltre il 70% del traffico IP totale entro la fine del 2022, rispetto a circa il 50% del 2019. Questo spostamento verso un maggiore utilizzo delle reti mobili ha delle implicazioni significative anche per l'utilizzo energetico delle reti di trasmissione dati, date le intensità elettriche (kWh/GB) notevolmente più elevate delle reti mobili rispetto alle reti fisse. Le reti mobili si stanno rapidamente spostando dalla vecchia tecnologia 2G e 3G a 4G e 5G che sono più efficienti. Entro il 2022, si prevede che le reti 4G e 5G trasporteranno insieme l'83% del traffico mobile, rispetto a meno dell'1% del 2G. Le reti 4G sono circa cinque volte più efficienti dal punto di vista energetico del

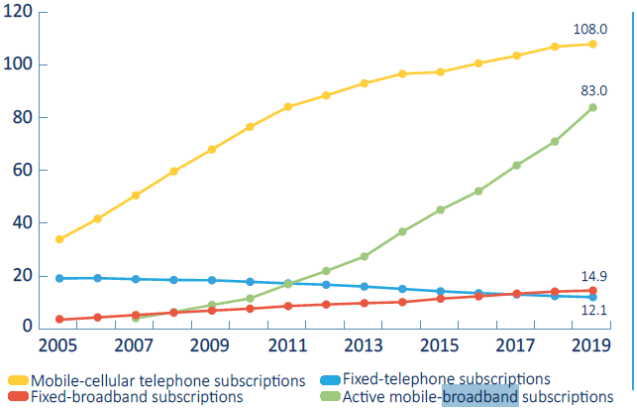


Figura 3.39: Evoluzione delle sottoscrizioni delle reti mobile e fisse negli anni. [60]

3G e 50 volte più efficienti del 2G. Gli impatti complessivi di energia ed emissioni del 5G sono ancora incerti; un'antenna 5G consuma attualmente circa tre volte più elettricità di un'antenna 4G ma, le funzionalità di risparmio energetico come la modalità di sospensione potrebbero ridurre il divario al 25% entro il 2022. I fornitori di infrastrutture di rete e gli operatori prevedono che il 5G potrebbe essere da 10 a 20 volte più efficiente dal punto di vista energetico del 4G entro il 2025-30.

Oltre ai consumi della rete mobile ci sono quelli degli smartphone stessi. Molti studi hanno analizzato i consumi di questi device al fine di allungare la durata della batteria e al fine di produrre componenti più efficienti e ridurre l'impatto ambientale. Anche per gli smartphone, come per i Laptop e i Desktop, i consumi dipendono sia dai componenti Hardware che dal Software.
Gli smartphone si trovano per lunghi periodi in sospensione, stato in cui il telefono non è usato effettivamente ma consuma perchè deve rimanere connesso alla rete per essere in grado di ricevere chiamate, messaggi SMS, ecc..

In generale i dispositivi più recenti in termini di sistema operativo e componenti hardware mostrano un consumo significativamente inferiore rispetto ai dispositivi meno recenti [61].

Per quanto riguarda il **Software**, esso non consuma direttamente ma influenza il consumo energetico dell'hardware sottostante. Inoltre l'impatto di diverse configurazioni della stessa applicazione provoca un comportamento di consumo energetico diverso [61].

Discorso simile per i **tablet**, che però hanno uno share del 64% (valore 2020), valore molto più basso rispetto a quello degli smartphone. Anche per i tablet il trend è in forte crescita, infatti si stima un incremento del 21% rispetto al 2019.

Raspberry

Raspberry Pi è un computer a scheda singola, un intero ecosistema hardware raccolto in un'unica board. Nasce in Inghilterra per favorire la diffusione della programmazione e della cultura informatica ma il suo incredibile successo ne ha svelato miriadi di altri utilizzi. Viene utilizzato come piattaforma per modellare il "cloud computing", per il monitoraggio e l'automazione della casa, per fornire computing a basso costo ai paesi in via di sviluppo.
La scheda è stata progettata per ospitare sistemi operativi basati sul kernel Linux e gira con un sistema operativo appositamente dedicato, chiamato Raspberry Pi OS. Raspberry Pi supporta un alimentazione USB con una potenza di 5 Volt.
Per un normale utilizzo, Raspberry Pi consuma solo 2,25W senza display e in "idle" circa 1,32W [62]; consumi molto più bassi rispetto a PC, Smartphone e Tablet (Figura 3.40) [63].

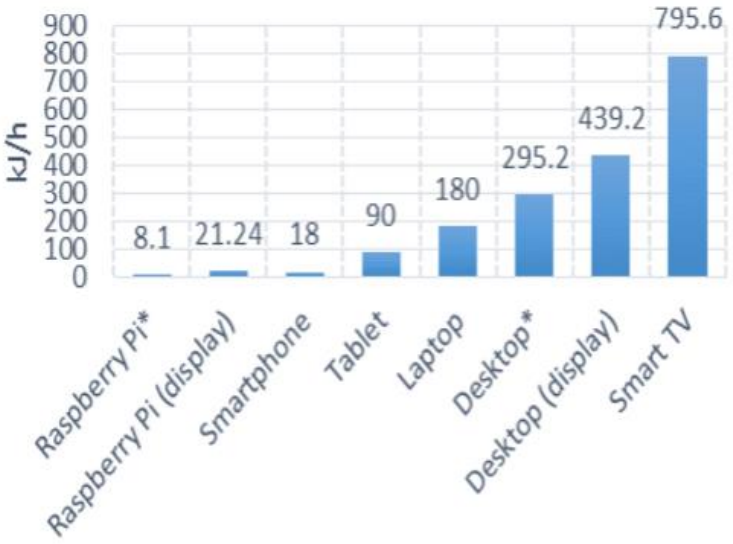


Figura 3.40: Confronto del consumo energetico di Raspberry Pi con altri device [63]

Scendendo un po' più nel dettaglio del consumo di Raspberry, in [64] è stato studiato un modello per stimare il consumo della SBC basato solo su utilizzo di CPU e della rete (Wi-Fi o Ethernet). Le misure fatte hanno evidenziato una relazione lineare tra Consumo e workload CPU (Figura 3.41).

La Raspberry viene sempre più impiegata in scenari *Edge-IoT*. Sulla SBC quindi vengono fatti girare vari container. L' utilizzo di tecnologie di virtualizzazione con

container su SBC producono un impatto quasi trascurabile in termini di consumi e prestazioni rispetto alle esecuzioni native. Questo risultato rimane valido anche durante l'esecuzione di diverse istanze virtualizzate contemporaneamente [62].

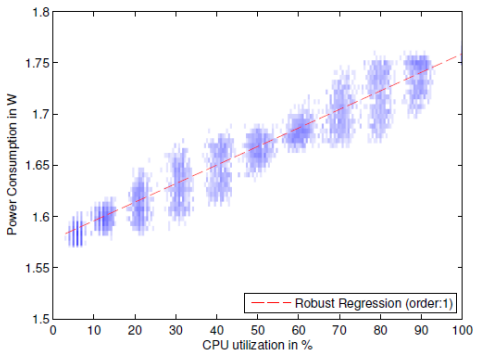


Figura 3.41: Relazione tra consumo e workload CPU in una Raspberry Pi [64]

Capitolo 4

Analisi del consumo di potenza

4.1 Le scelte fatte

Esistono vari metodi per loggare i consumi, tra questi:

- `Smart Plug`: vedi 4.1
- `Multimetro`: è uno strumento utilizzato per misurare due o più valori elettrici, principalmente la tensione (volt), la corrente (ampere) e la resistenza (ohm). Si tratta di uno strumento diagnostico standard per i tecnici in ambito elettrico ed elettronico.
- `Pinza amperometrica`: è uno strumento di misura portatile che si utilizza per misurare la corrente, la tensione, la potenza, la capacità, la resistenza. La pinza amperometrica (a differenza del multimetro) è in grado di misurare correnti venendo semplicemente "agganciata" attorno al conduttore: questa infatti rileva l'intensità del campo elettromagnetico presente attorno al cavo e la trasforma in un segnale elettrico inviato al misuratore.
- `Tester USB`: è un dispositivo che viene inserito nelle porte USB (sia USB-A, micro-USB e USB-C) e misura tensione, corrente, potenza, energia e temperatura. Sono normalmente di dimensioni estremamente limitate, a volte sono dotati di connessione bluetooth e permettono di visualizzare i dati in real-time sul display della quale sono dotati (oltre che consentire l'esportazione dei dati rilevati). Hanno un range di misura limitato.
- `Altro` - Sul mercato esistono anche sistemi professionali centralizzati, solitamente molto costosi, che permettono di misurare il consumo energetico.

Le prese intelligenti

Le **prese intelligenti** sono dispositivi che vanno inseriti nella presa a muro, e ad esse vanno collegati i dispositivi che si intende "comandare" da remoto, per renderlo intelligente. Una volta collegata la smart plug all' app di gestione sullo smartphone, è possibile attivarla o disattivarla da remoto sfruttando la stessa rete (o utilizzando la connessione dati sullo smartphone). Alcune prese intelligenti permettono di visualizzare i consumi energetici del prodotto attaccato alla presa e anche di vedere il tempo di accensione dello stesso. Si tratta di una caratteristica davvero molto interessante in quanto è possibile capire quanto consuma un determinato prodotto e trovare un sistema per ottimizzarne i consumi.

Di seguito le funzionalità degli smart plug:

- `Accesso remoto` - Controlla i dispositivi connessi allo Smart Plug ovunque ci sia Internet utilizzando le varie app disponibili sullo smartphone
- `Pianificazione` - Programma lo Smart Plug per fornire automaticamente energia a seconda della necessità, come impostare le luci al crepuscolo e spegnerle all'alba.
- `Modalità a distanza` - Attiva o disattiva i dispositivi da remoto.
- `Amazon Echo Controllo Vocale` - Amazon Echo (venduto separatamente) consente di controllare i dispositivi connessi allo Smart Plug utilizzando comandi vocali.
- `Domotica` - Creare degli scenari di utilizzo che permetteranno di far funzionare più prese contemporaneamente, per esempio accendendole tutte insieme o spegnendole in un solo click.
- `Consumi` - Tracciare i consumi dei dispositivi collegati

Sul mercato esistono vari produttori di prese intelligenti che permettono il tracciamento dei consumi. Tramite le app degli stessi produttori, o app di terze parti (HomeKit, Kasa) però, è possibile solo visionare i grafici dei consumi ma non esportarli in formati come CSV o TXT, per successive analisi. La possibilità di logging è invece messa a disposizione da librerie Python di terze parti che forniscono le API che permettono anche il controllo dei dispositivi da remoto tramite Internet. Tra queste c'è la libreria **Meross-IoT** [65] che supporta le prese Meross MSS310 [1] (Smart plug con "energy monitor") [66].

[1] https://www.meross.com/product/6/article/

4.1.1 Gli script e le librerie

Logging consumo prese - libreria *MerossIoT*

Per loggare i consumi delle prese Meross è stato utilizzato uno script Python [2] (compatibile con una versione Python 3.7+) e la libreria *Meross-IoT*. Tale libreria, è facilmente scaricabile con il comando

```
pip install meross-iot --upgrade
```

Meross-IoT [3] fornisce le API per gestire vari dispositivi Meross. Le principali funzionalità sono:

- `Elencare device`: Listare i dispositivi connessi all' account Meross, anche filtrandoli secondo alcuni criteri (es "device_type", "device_class", "online_status"..)
- `Commutare gli interruttori intelligenti`: Cambiare lo stato dei device da acceso a spento e viceversa
- `Controllare le lampadine intelligenti`: Gestire le impostazioni del colore RGB, nonché la luminosità e la temperatura del colore
- `Controllo degli apriporta del garage`: Permette di capire lo stato della porta (aperta/ chiusa) e simula la pressione del pulsante apriporta del garage.
- `Lettura di sensori`: Alcuni dispositivi Meross sono dotati di sensori. Alcuni device (come il sensore di temperatura e umidità) sono sensori di sola lettura; altri, come le prese l'MSS310, la valvola Termostatica o gli apri garage sono invece attuatori che offrono alcune capacità di lettura dei dati (es. consumo di potenza)
- `Azionare Termostati`: Permette di accendere/ spegnere il termostato e impostare una temperatura d'azione

Associazione delle prese

L'associazione di un nuovo dispositivo Meross può essere descritto come segue [65]:

[2] https://www.python.org/

[3] Le informazioni contenute in questo capitolo sono recuperate dalla documentazione presente sul sito *https://pypi.org/project/meross-iot/*

- L'utente mette il dispositivo in modalità di associazione (premendo e tenendo premuto il pulsante hardware sul dispositivo)
- L'utente utilizza l'APP sullo smartphone (Android o Apple) per configurare il dispositivo e associarlo a un account Meross
- La spina si connette al Wifi, quindi si connette al broker Meross MQTT

Una volta che l'utente mette il dispositivo in "modalità di associazione", il dispositivo si mette in modalità Access Point, configurando una rete Wifi aperta. Il nome di questa rete inizia come "Meross_<STR1>_<STR2>". Ciò consente all'App di riconoscere tutti i dispositivi Meross disponibili per l'associazione semplicemente scansionando tutte le reti wifi e filtrando tramite il prefisso SSID. Successivamente l'app si connette al Wifi del dispositivo da accoppiare e ottiene un indirizzo IP tramite DHCP. A questo punto, l'APP manda due POST HTTP in sequenza alla spina:

1. Lo scopo della prima richiesta è dire alla presa a quale broker MQTT connettersi e quali sono le credenziali che deve usare per autenticarsi ad esso.
2. Lo scopo della seconda richiesta è quello di impostare i parametri della connessione Wifi domestica per connettere quindi la presa ad internet.

Alla fine la presa si riavvia e tenta di connettersi alla rete Wifi e, in caso di successo, anche al broker MQTT.

Architettura Meross MQTT

La maggior parte delle comunicazioni tra l'app Meross e i dispositivi avviene tramite un broker MQTT 4.1.

Dall'immagine 4.1, vediamo che esistono quattro tipi diversi di topic:

- `/appliance/<device_uuid>/subscribe`
  Rappresenta il topic da cui l'appliance estrae i comandi da eseguire. E' specifico per ogni dispositivo Meross visto che nella chiamata viene indicato *device_uuid* che è univoco per ogni device.
- `/appliance/<device_uuid>/publish`
  È il topic con cui l'appliance pubblica gli eventi (notifiche push). E' specifico per ogni dispositivo Meross.
- `/app/<user_id>/subscribe`
  L'App Meross si iscrive a questo topic per aggiornare il suo stato man mano che gli eventi si verificano sul dispositivo fisico. E' specifico per *user_id*.

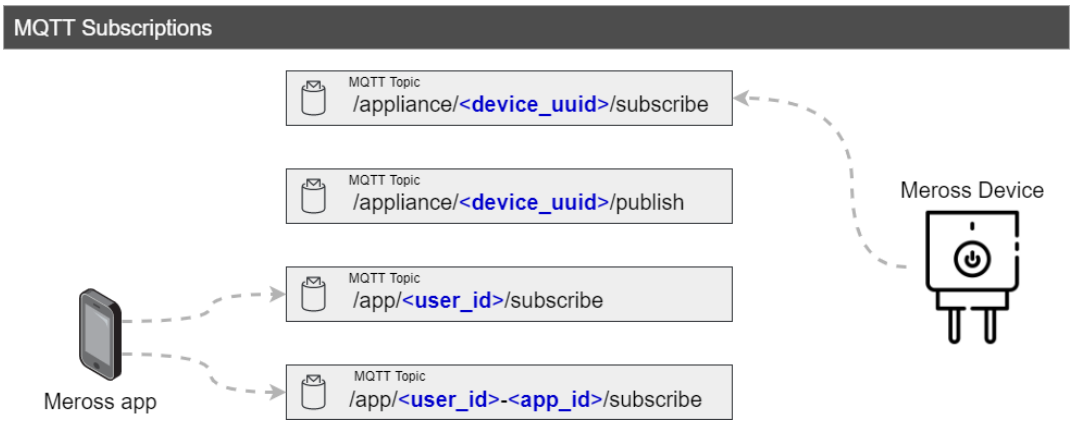


Figura 4.1: MTTQ subscription

[65]

- `/app/<user_id>-<app_id>/subscribe`
  È il topic a cui si iscrive l'App Meross e viene utilizzato dall'app per ricevere la risposta ai comandi inviati all'appliance.

Nelle immagini 4.2 e 4.3 vediamo come vengono utilizzati i vari topic sopra descritti.

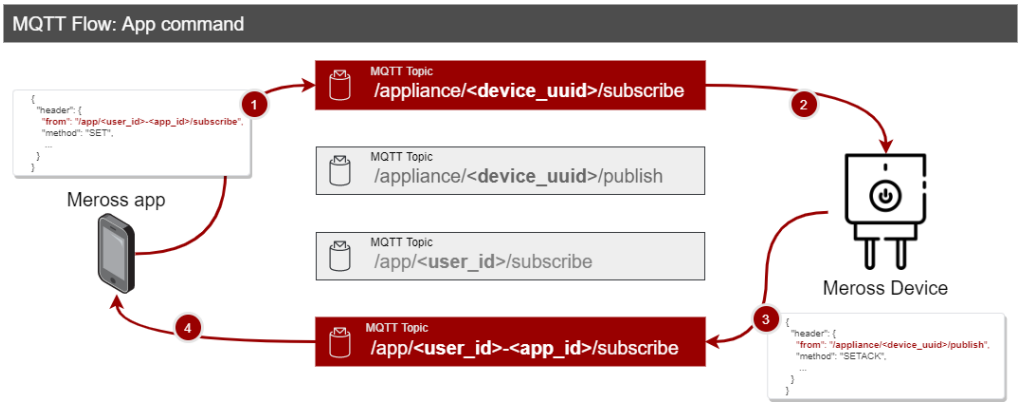


Figura 4.2: Il flusso dei comandi tra app e device

[65]

Script per il logging del consumo

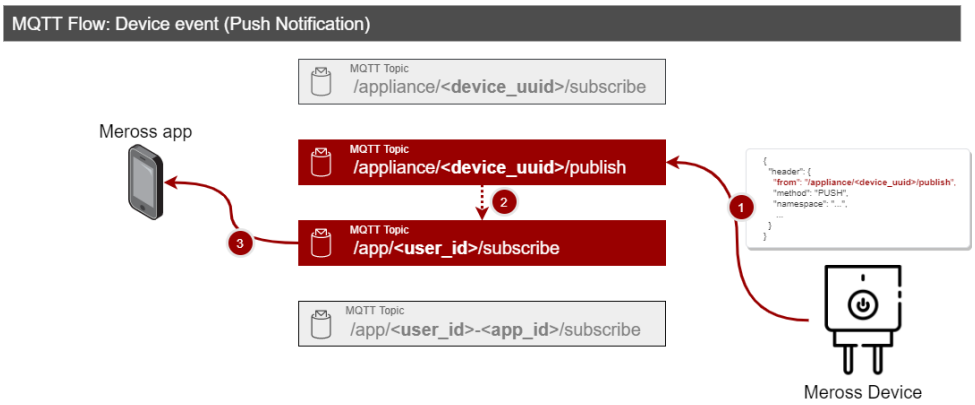


Figura 4.3: Flusso degli eventi (notifiche push) dei device [65]

Listing 4.1: Funzione per il logging del consumo con libreria Meross-IoT

```
EMAIL = os.environ.get('email@yahoo.it') or "email@yahoo.it"
PASSWORD = os.environ.get('password') or "password"

async def main():
    global http_api_client
    global manager
    global f
    dt = datetime.datetime.now()
    # Setup the HTTP client API from user-password
    http_api_client = await MerossHttpClient.async_from_user_password(email=EMAIL, password=PASSWORD)
    # Setup and start the device manager
    manager = MerossManager(http_client=http_api_client)
    await manager.async_init()
    # Retrieve all the MSS310 devices that are registered on this account
    await manager.async_device_discovery()
    devs = manager.find_devices(device_class=ElectricityMixin)
    if len(devs) < 1:
        print("Nessun dispositivo trovato")
    else:
        for dev in devs:
            await dev.async_update()
        #Saving power information of all MSS310 found online
        for dev in devs:
            instant_consumption = await dev.async_get_instant_metrics()
            with open("consumi.txt",'a+') as f:
```

```
26              f.write(dev.internal_id +", "+ dev.name +","+ str(dt.
    strftime('%Y-%m-%d')) +' '+ str(dt.strftime('%H:%M')) + f", {
    instant_consumption.power}"+ '\n')
27              f.close()
28              # Close the manager and logout from http_api
29      manager.close()
30      await http_api_client.async_logout()
31
32
```

Lo script 4.1 ricava la potenza consumata nell'istante in cui viene richiamata la funzione stessa, quindi per avere misure periodiche è necessario impostare una utility che consenta di pianificare l'esecuzione automatica periodica di attività (per esempio su OSX / Linux è possibile usare **crontab** e per Windows **Utilità di Pianificazione**. Nella figura 4.4 un esempio di file TXT generato dallo script 4.1 richiamato a intervalli periodici di 5 minuti.

Separati dalla , abbiamo i seguenti campi:

- *device_uuid*: ID univoco di ogni device
- *Nome*: Nome scelto dall'utente durante la configurazione del device nell' App Meross
- *Data e Ora*
- *Consumo di Potenza*

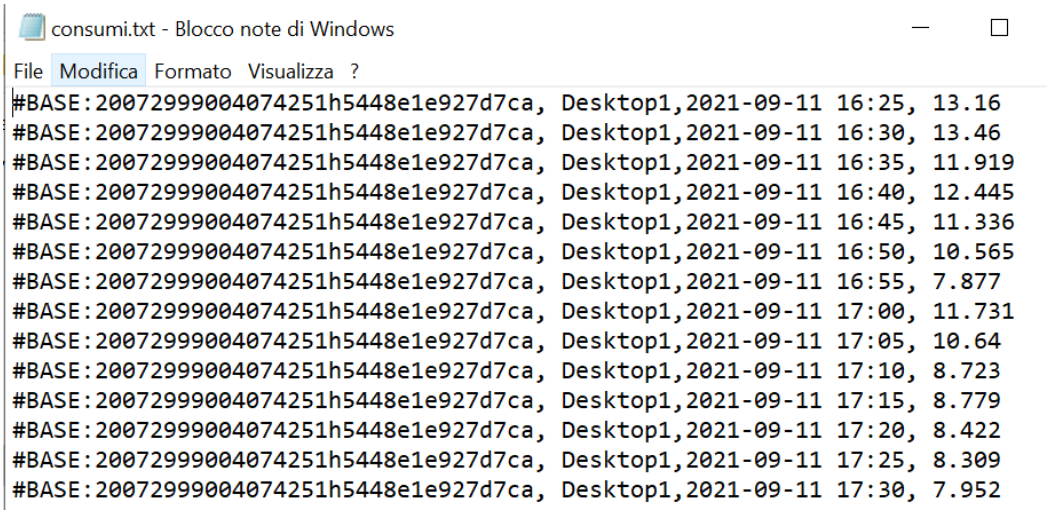


Figura 4.4: File TXT contenente i consumi, generato dallo script 4.1

Logging del workload di un sistema

Insieme al consumo è necessario misurare anche il carico del sistema per:

- Verificare eventuali correlazioni tra il consumo e il workload del sistema
- Trovare un modello di predizione per stimare il consumo

Libreria *psutil*

Psutil *"python system and process utilities"* è una libreria cross platform (Linux, Windows, MacOS, FreeBSD, OpenBSD, NetBSD, Sun Solaris) con la quale è possibile ricavare informazioni sull' utilizzo delle risorse di sistema (CPU, memoria, dischi, rete..). È particolarmente utile per il monitoraggio dei sistemi, la profilazione, la limitazione delle risorse dei processi e la gestione dei processi in esecuzione. Implementa molte funzionalità offerte dai classici strumenti da riga di comando UNIX come *ps, top, iotop, lsof, netstat, ifconfig, free* e altri. Psutil è supportata da versioni di Python 2.6, 2.7 e 3.4+ [67].

Script per il logging del worload

Listing 4.2: Funzione per il logging del worload di un sistema con libreria psutil

```
1 async def main():
2     dt = datetime.datetime.now()
3     svmem = psutil.virtual_memory()
4     with open("sistema.txt",'a+') as f:
5       f.write(str(dt.strftime('%Y-%m-%d')) +' '+ str(dt.strftime('%H:%
    M'))+","+ str(psutil.cpu_percent(1)) +","+str(svmem.percent)+ '\n
    ')
6       f.close()
```

Lo script 4.2 è stato richiamato periodicamente e ha generato un file TXT come quello nella figura 4.5 e contiene i seguenti dati:

- `Data e Ora`
- `Workload della CPU`
- `Workload della RAM`

4.2 Analisi dati

Le misure sono state fatte in alcuni ambienti residenziali e in una azienda utilizzando gli script descritti nella sezione precedente. Il consumo intrinseco delle prese intelligenti viene considerato irrilevante.

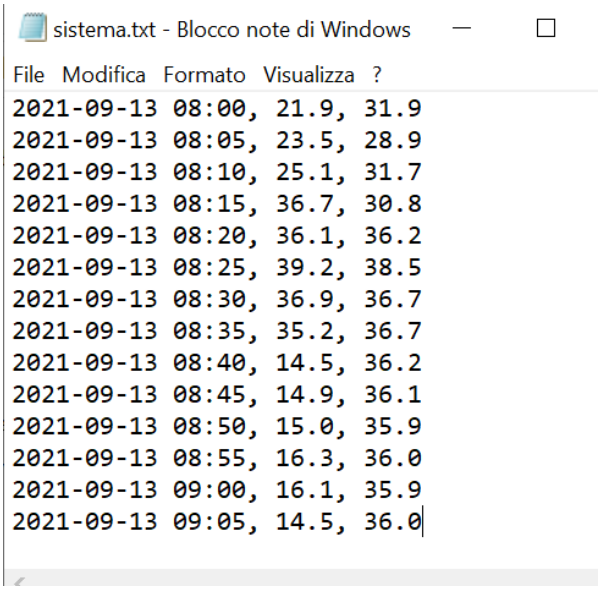


Figura 4.5: File TXT contenente i dati di workload del sistema misurato, generato dallo script 4.2

Tipo	Numero
Desktop	18
Laptop	14
Altri device	30

Tabella 4.1: Tabella riassuntiva dei dispositivi misurati

4.2.1 Relazione tra consumo e workload

Analizzando i dati ricavati con lo script 4.1.1 e mettendoli in relazione sono state ricavate le osservazioni che saranno esposte in questa sezione.

Dai grafici 4.6, 4.7, 4.8 e 4.9, a livello visivo si può notare come i vari picchi di consumo si trovano in corrispondenza dei vari picchi di CPU. Si intuisce quindi che potrebbe esistere una relazione tra il workload della CPU e la potenza consumata. Per questo motivo, con dei grafici a dispersione, si sono correlate le due grandezze per capire se effettivamente, ciò che si è visto "ad occhio" abbia delle conferme più scientifiche.

Dai grafici a dispersione si evince una relazione positiva, infatti all'aumentare della variabile Consumo (Asse X), aumenta anche la variabile CPU (Asse Y).

Nei grafici è rappresentata anche la retta di regressione e il corrispondente

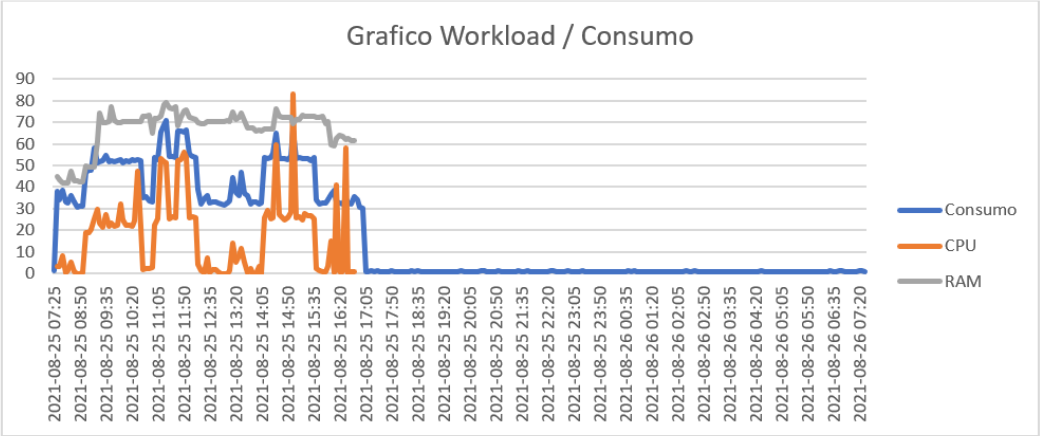


Figura 4.6: Grafico Consumo-CPU-RAM di un Desktop1

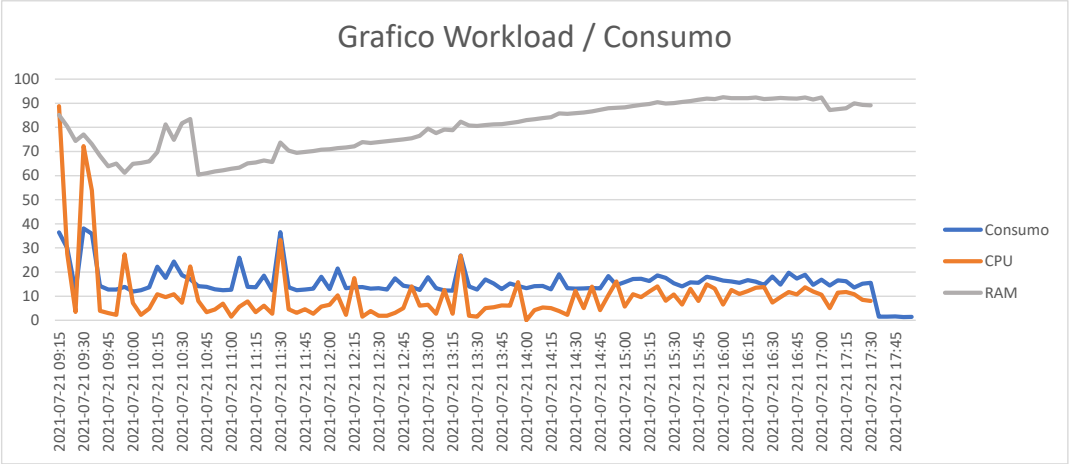


Figura 4.7: Grafico Consumo-CPU-RAM di un Desktop2

valore di R^2. Si nota che il valore di R^2 è abbastanza alto da poter considerare una correlazione tra le due variabili studiate; infatti più è grande il valore di R^2, più il modello ha un alto potere predittivo e migliore è la capacità delle variabili esplicative di prevedere i valori della variabile dipendente.

Si deve anche considerare che gli outlier, introdotti da degli errori di misura, influenzano parecchio il modello lineare e fanno diminuire il valore di R^2.

Dai grafici a dispersione si può anche osservare come, i vari sistemi consumano anche con il workload della CPU pressochè tendente allo 0. Ad esempio il Desktop 4.6 ha un consumo di circa 30W con il processore a 0%.

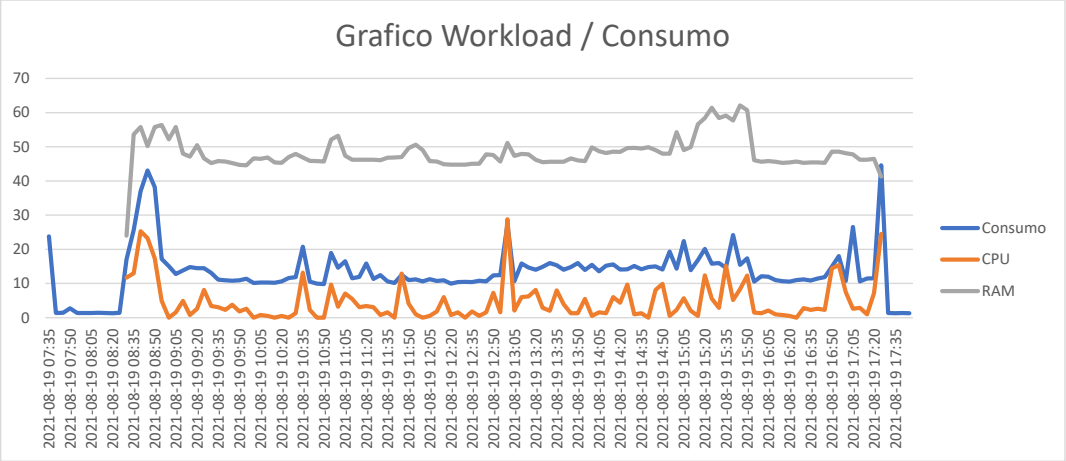


Figura 4.8: Grafico Consumo-CPU-RAM di un Desktop3

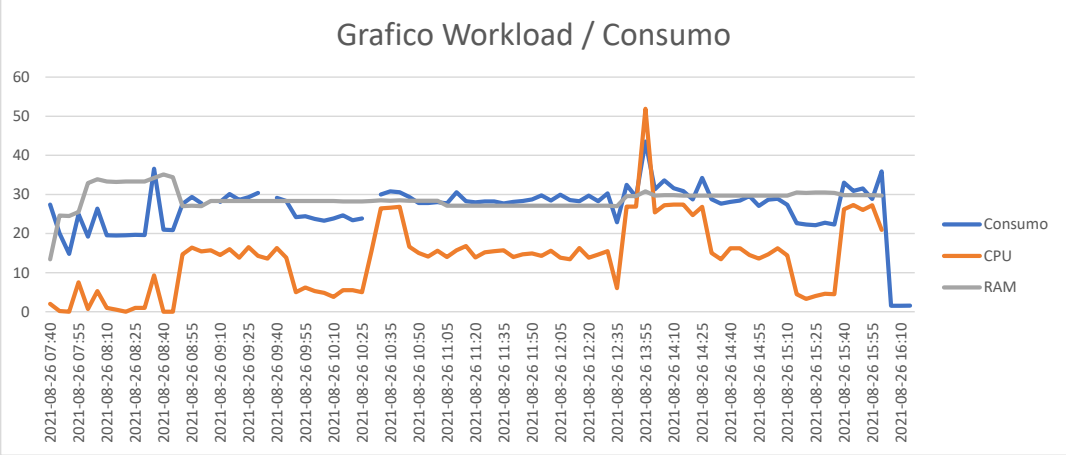


Figura 4.9: Grafico Consumo-CPU-RAM di un Laptop1

4.2.2 Osservazione sui consumi

Dai dati rilevati si possono fare le seguenti osservazioni:

- `I sistemi risultano sottoutilizzati durante tutta il loro utilizzo.` Nei grafici di esempio riportati (4.6, 4.7, 4.8 e 4.9) si vede come il workload sia di CPU che RAM risultano essere sempre (a meno di alcuni picchi) su valori molto inferiori al carico massimo del 100%. Per esempio nel grafico 4.9, la CPU resta sempre al di sotto del 30% a meno di un picco; così anche

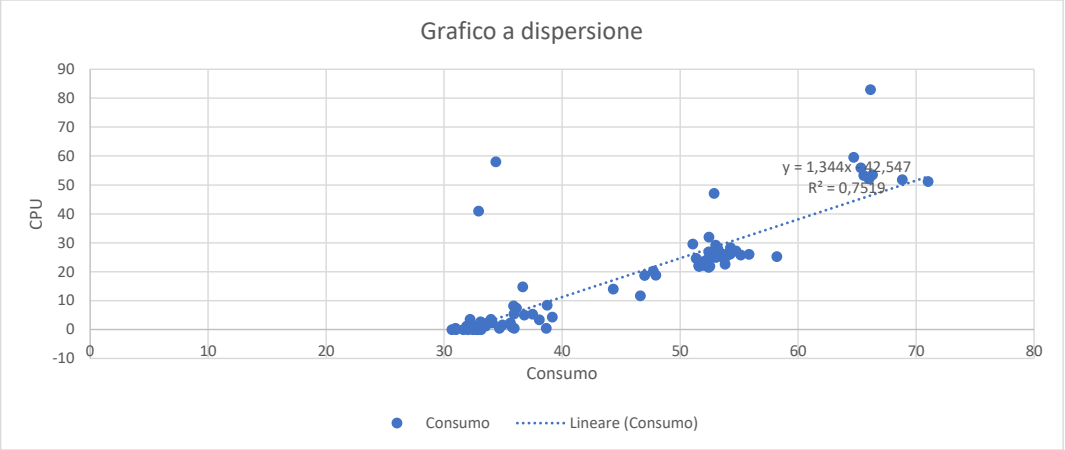


Figura 4.10: Grafico a dispersione del Desktop1 4.6

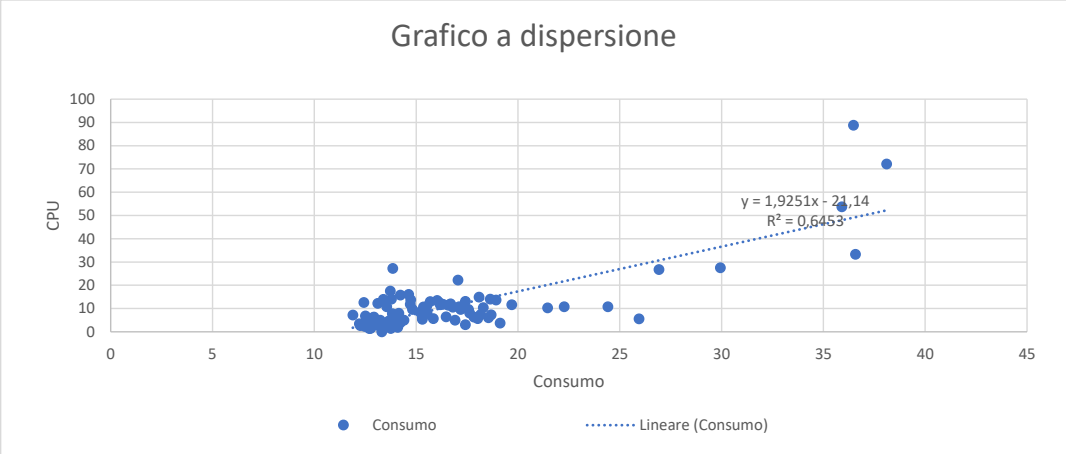


Figura 4.11: Grafico a dispersione del Desktop2 4.7

per la RAM che rimane compresa tra i 30% e i 35% in tutto l'intervallo di misurazione.

- `PC più datati consumano di più`
  Come visibile dal grafico 4.15 i Desktop che consumano di più sono i meno recenti: 1. *Desktop3i5* (2010) 2. *Desktop1Pentium* (2009) 3. *Desktop2Pentium*(2011). Discorso analogo per i Laptop (Figura 4.14). Questo a sostegno del fatto che negli anni c'è stato un efficientamento dei sistemi in termini di consumo.

- `Utilizzi diversi portano a consumi diversi`

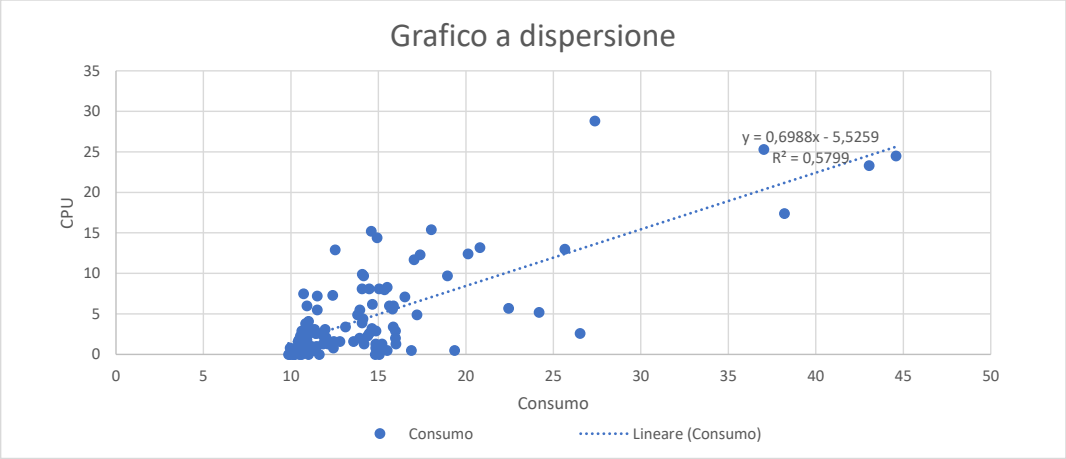


Figura 4.12: Grafico a dispersione del Desktop3 4.8

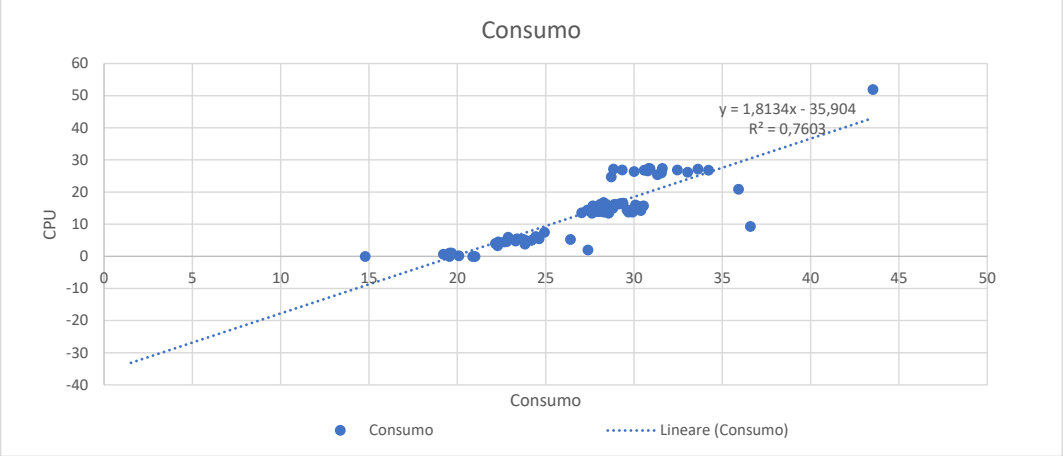


Figura 4.13: Grafico a dispersione del Laptop1 4.9

Oltre ai diversi pattern di utilizzo in ambito lavorativo e casalingo, sistemi uguali hanno un consumo diverso. Come si può notare dalla figura 4.16 consumi di Desktop uguali hanno una distribuzione gaussiana diversa indicando proprio un consumo e quindi un utilizzo differente.

- `Consumi`
  I computer consumano anche con CPU a 0, per esempio nel grafico 4.13 il consumo a CPU scarica è di circa 20W.
  I computer hanno un consumo, seppur minore anche in stand by e un consumo

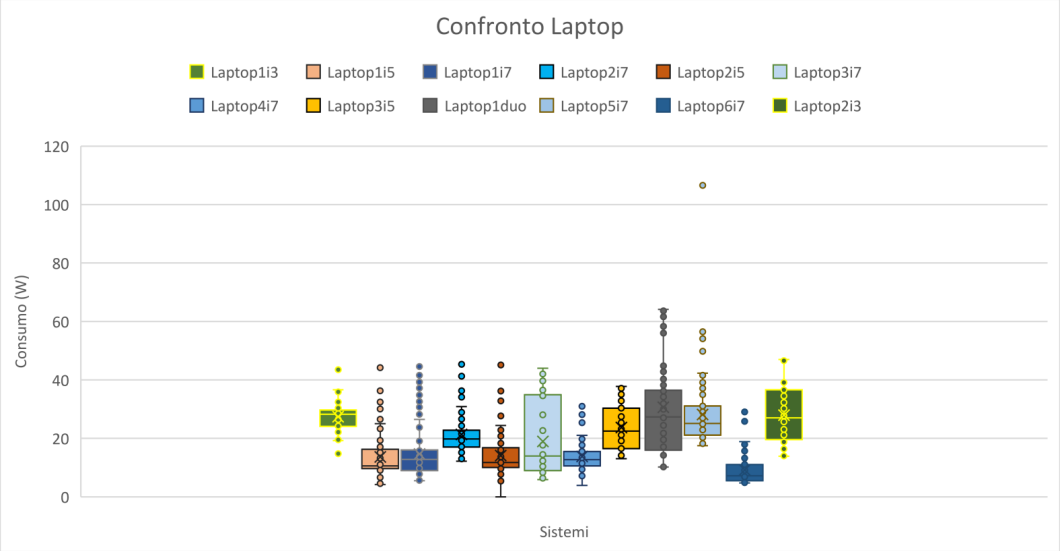


Figura 4.14: BoxPlot dei Laptop contenuti nella tabella 4.2;
Nelle gradazioni di `blu` gli i7, `arancione` gli i5, `verde` gli i3, `grigio` gli Intel Core 2 DUO.

Identificativo	Processore	Anno
Laptop1i3	Intel(R) Core(TM) i3 CPU 350M @ 2,27GHz x4	Q1 2010
Laptop1i5	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz	Q2 2017
Laptop1i7	Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz	Q2 2017
Laptop2i7	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz	Q4 2016
Laptop2i5	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz	Q2 2017
Laptop3i7	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz	Q2 2018
Laptop4i7	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz	Q4 2016
Laptop3i5	Intel Core i5-4210M 2,60GHz	Q2 2014
Laptop1duo	Intel(R) Core(TM) 2 DUO CPU T6670 @ 2.20GHz	Q2 2009
Laptop5i7	Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz	Q2 2014
Laptop6i7	Intel Core i7-8550U 1,8 GHz	Q2 2017
Laptop2i3	Intel(R) Core(TM) i3-3120M CPU @ 2.50GHz	Q1 2012

Tabella 4.2: Tabella con l'elenco dei Laptop

irrisorio anche da spenti.

- `Consumi Desktop e Laptop`

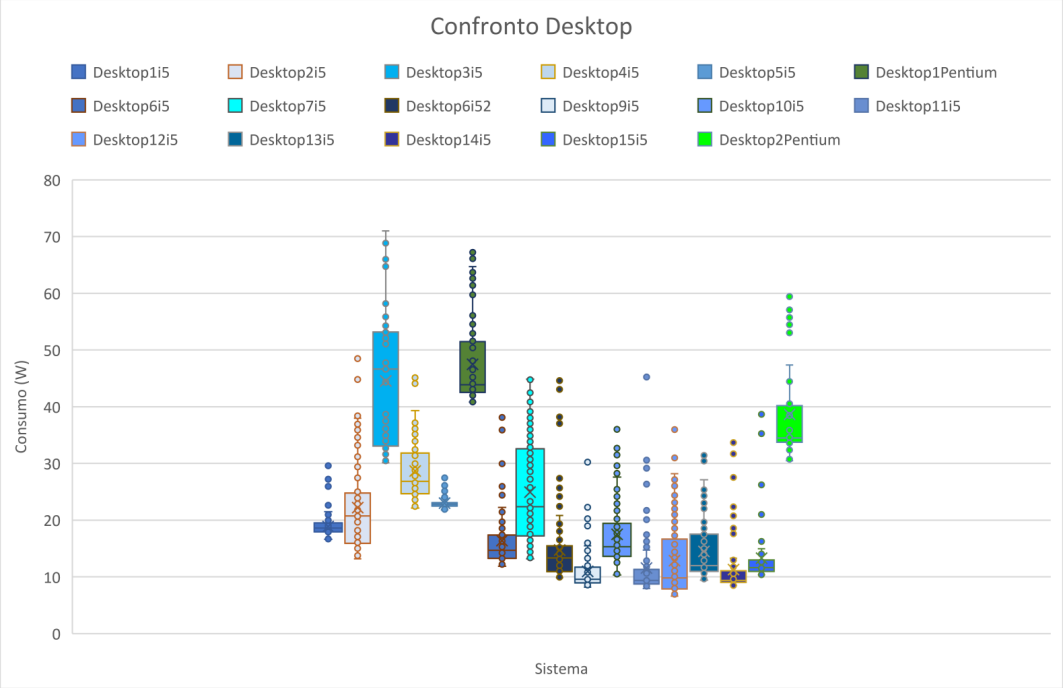


Figura 4.15: BoxPlot dei Desktop contenuti nella tabella 4.3;
Nelle gradazioni di `blu` gli i5, `verde` i Pentium.

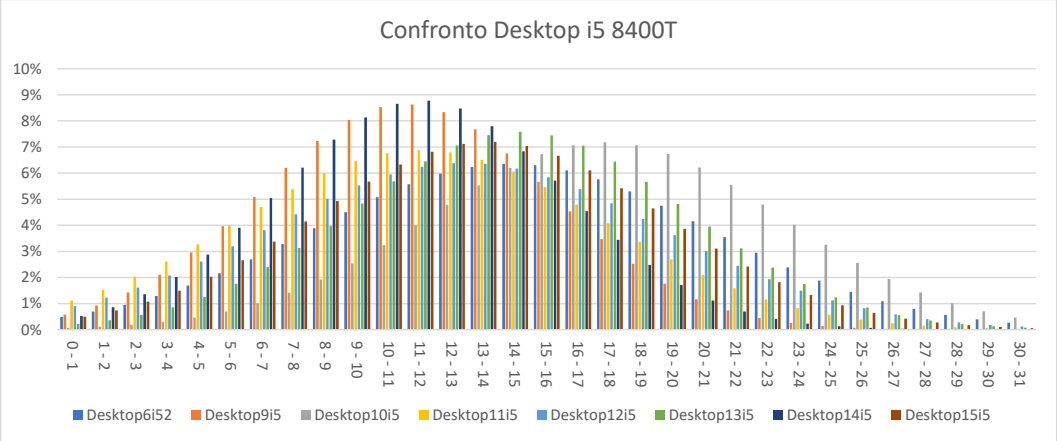


Figura 4.16: Gaussiana di confronto di diversi Desktop con processore i5 8400T

Confrontando un Desktop e un Laptop si nota come il portatile abbia un consumo inferiore. Per esempio nella Figura 4.17 entrambi i sistemi hanno un

Identificativo	Processore	Anno
Desktop1i5	Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz x4	Q2 2015
Desktop2i5	Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz	Q2 2015
Desktop3i5	Intel(R) Core(TM) i5-650 3,20 GHz	Q1 2010
Desktop4i5	Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz	Q2 2012
Desktop5i5	Intel(R) Core(TM) i5-8500T CPU @ 2.10GHz	Q2 2018
Desktop1Pentium	Intel Pentium(R) Dual-Core CPU E5400 @ 2.70GHz	Q2 2009
Desktop6i5	Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz	Q2 2015
Desktop7i5	Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz	Q2 2015
Desktop6i52	Intel(R) Core(TM) i5-8400T CPU @ 1.70GHz	Q2 2018
Desktop9i5	Intel(R) Core(TM) i5-8400T CPU @ 1.70GHz	Q2 2018
Desktop10i5	Intel(R) Core(TM) i5-8400T CPU @ 1.70GHz	Q2 2018
Desktop11i5	Intel(R) Core(TM) i5-8400T CPU @ 1.70GHz	Q2 2018
Desktop12i5	Intel(R) Core(TM) i5-10400T CPU @ 2.00GHz	Q2 2020
Desktop13i5	Intel(R) Core(TM) i5-8400T CPU @ 1.70GHz	Q2 2018
Desktop14i5	Intel(R) Core(TM) i5-8400T CPU @ 1.70GHz	Q2 2018
Desktop15i5	Intel(R) Core(TM) i5-8400T CPU @ 1.70GHz	Q2 2018
Desktop2Pentium	Intel Pentium CPU G850 @ 2.90GHz	Q2 2011

Tabella 4.3: Tabella con l'elenco dei Desktop

processore della stessa famiglia (Intel i5) rilasciati pressochè nello stesso periodo. E' possibile notare come, il Laptop che nelle misurazioni ha un workload medio della CPU doppio rispetto al Desktop (LaptopCPU= 37,88%; DesktopCPU= 18,36%) ha un consumo medio leggermente superiore al Desktop.

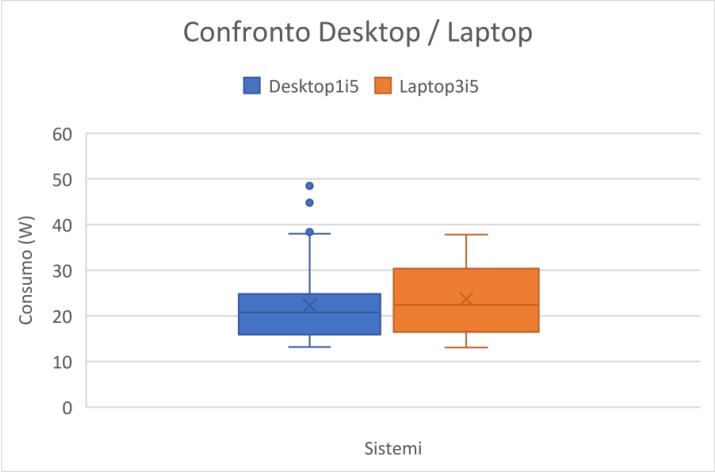


Figura 4.17: Boxplot del consumo di un Desktop (in blu) e un Laptop (in arancione) con processore Intel i5 sviluppati in anni adiacenti

Capitolo 5

Consumo di potenza di K3s

5.0.1 Introduzione

Nella vita odierna siamo circondati da molti device. Dispositivi tra di loro autonomi progettati con le risorse adeguate a svolgere i compiti per la quale sono stati costruiti. Una visione logica di questa situazione è schematizzata nella Figura 5.1

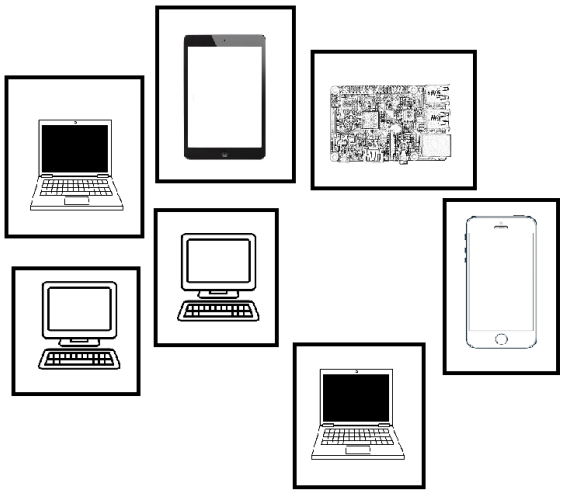


Figura 5.1: Una visione logica dei dispositivi che ci circondano. I device sono tra di loro indipendenti e progettati con le risorse necessarie per essere autonomi.

Questa "indipendenza" porta però i device ad avere risorse che, per la maggior parte del tempo, risultano essere sottoutilizzate. Come abbiamo visto nel capitoli precedenti (3 e 4) anche solo il fatto che i dispositivi siano connessi ad una presa elettrica producono un consumo di potenza che genera un impatto ambientale.

Il consumo totale dei device di questo panorama sarà:

$$C_{tot1} = \sum_{i=1}^{N} C_i$$

dove C_i è il consumo dei singoli dispositivi.

Una possibile soluzione (quella considerata in questa tesi) potrebbe essere quella di condividere le risorse tra device (visione logica visibile in Figura 5.2) in modo da valutare se effettivamente si ottiene un risparmio energetico.

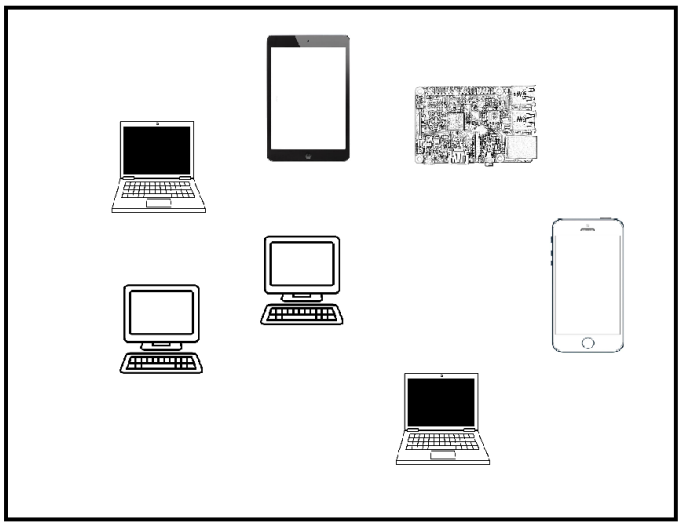


Figura 5.2: Una visione logica dei dispositivi con risorse condivise. I device condividono le risorse tra di loro grazie ad un software (nel nostro caso `K3s`).

Per poter implementare tale soluzione in un ambiente edge c'è la necessità di utilizzare un software aggiuntivo da installare sui dispositivi. Esistono varie possibilità: quella da noi presa in considerazione utilizza **K3s**. In questa visione il consumo totale sarà:

$$C_{tot2} = \sum_{i=1}^{N} (C_i + C_{sharing})$$

dove C_i è il consumo dei singoli dispositivi e $C_{sharing}$ è il consumo del software che permette la condivisione di risorse.

5.0.2 Analisi consumo di K3s

K3s è un software che gira sui vari dispositivi e concorre quindi al consumo totale del sistema sulla quale è installato.

Per misurare il consumo di K3s si è utilizzata una VM con 4 vCPU, 8 GB di RAM e 15 GB di disco e il software `sysstat` [1].
Sysstat [69] è un software gratuito disponibile nelle diverse distribuzioni Linux che, oltre alle altre funzionalità che per brevità non verranno citate, permette di monitorare e successivamente estrapolare in formati come TXT, CSV le statistiche della CPU.

Dalle misurazioni svolte con periodicità di 10 minuti, è risultato che il nodo master consuma in media 5,096% di CPU, il worker node invece 1,468%.

5.0.3 Comparazione del consumo con e senza K3s

Come esposto nel capitolo 4 sono state svolte delle misure (con la visione logica presente nella Figura 5.1) su dei sistemi durante il loro normale utilizzo quotidiano. Per stimare i consumi della soluzione con risorse condivise con K3s (Figura 5.2) è stato utilizzata la retta di regressione dei vari grafici a dispersione. Questo modello stima il consumo totale del sistema tramite il workload della CPU. Spostando parte dei carichi da dispositivi meno potenti a quelli più potenti, si ha un workload che viene riscalato a seconda delle diversità di potenza dei dispositivi convolti (per esempio spostando il 10% del carico da un Intel Pentium del 2011 ad un moderno Intel i7 di ultima generazione, avremo che il workload aggiuntivo su questa seconda postazione sarà sicuramente un carico proporzionale alla sua potenza; sicuramente inferiore al 10%). Per fare questo riscalamento si è utilizzato il valore "Average CPU Mark" presente sul sito `CpuBenchmark` [70]. Tramite quindi il modello lineare trovato sperimentalmente con misure sul campo, il consumo di CPU di K3s e questo riproporzionamento dei carichi con *CpuBenchmark* si è stimato il consumo con la visione a risorse condivise con K3s.

Di seguito si riportano alcuni esempi di spostamento carichi tra device.

ID sistema	CPU	CPUBenchmark
Desktop1Pentium	Intel Pentium(R) CPU E5400 @ 2.70GHz	914
Desktop14i5	Intel(R) Core(TM) i5-8400T CPU @ 1.70GHz	7317

Tabella 5.1: Caratteristiche dei due sistemi (*Desktop1Pentium* e *Desktop14i5*) sulla quale è stata fatta l'analisi presente in tabella 5.2

Dalle tabelle 5.2 e 5.6 si nota come lo spostamento di parte del carico da Desktop/Laptop meno potenti a quelli più performati si ha un risparmio nei consumi che aumenta con l'incremento del workload.

[1] Queste misure sono state fatte da un altro membro di questo team di progetto [68]

% Carico spostato	Consumo Iniziale	Consumo Finale	Risparmio
5	76,74	76,26	-0,48
15	76,74	72,68	-4,07
15	87,78	83,71	-4,07
20	98,81	92,95	-5,86
25	98,81	91,15	-7,65

Tabella 5.2: Analisi spostamento carico da *Desktop1Pentium* a *Desktop14i5*

ID sistema	CPU	CPUBenchmark
Raspberry	Quad-core ARM Cortex-A7 CPU 0.9 GHz	233
Desktop14i5	Intel(R) Core(TM) i5-8400T CPU @ 1.70GHz	7317

Tabella 5.3: Caratteristiche dei due sistemi (*Raspberry* e *Desktop14i5*) sulla quale è stata fatta l'analisi presente in tabella 5.4

Nella tabella 5.4 invece, vediamo che lo spostamento di task da una raspberry ad un PC molto più potente porta ad un aumento dei consumi. Questo perchè la Raspberry è un device a basso consumo.

% Carico spostato	Consumo Iniziale	Consumo Finale	Risparmio
5	30,48	31,91	1,43
15	30,48	34,75	4,27
20	41,51	47,21	5,7
25	52,55	59,67	7,12

Tabella 5.4: Analisi spostamento carico da *Raspberry* a *Desktop14i5*

ID sistema	CPU	CPUBenchmark
Laptop1i3	Intel(R) Core(TM) i3 CPU M350 @ 2,27GHz x4	1059
Desktop12i5	Intel(R) Core(TM) i5-10400T CPU @ 2.00GHz	10111

Tabella 5.5: Caratteristiche dei due sistemi (*Laptop1i3* e *Desktop12i5*) sulla quale è stata fatta l'analisi presente in tabella 5.6

% Carico spostato	Consumo Iniziale	Consumo Finale	Risparmio
5	87,6	88,22	0,63
15	87,6	83,76	-3,83
20	114,84	108,77	-6,07
25	142,08	133,78	-8,3

Tabella 5.6: Analisi spostamento carico da *Laptop1i3* a *Desktop12i5*

Capitolo 6

Conclusioni e futuri step

Il lavoro di tesi ha portato alla misurazione dei consumi di vari dispositivi e alla realizzazione di un modello con la quale si è stimato i consumi energetici dell' utilizzo del paradigma della `condivisione di risorse`: spostando dei task da dispositivi poco potenti a device più performanti si è rilevato un risparmio energetico. Tale risparmio si prevede possa riflettersi in modo positivo sulle emissioni di CO_2 e sul cambiamento climatico.
Visti i trend di penetrazione dell' ICT nella vita quotidiana e visto che l'efficientamento hardware ha ormai raggiunto il limite (*legge di Moore*), questa soluzione di condivisione di risorse tra dispositivi IoT potrebbe essere quindi una strategia utilizzabile per ridurre i consumi ed emissioni.

Si è inoltre rilevato:

1. una correlazione tra il workload della CPU e il consumo di potenza dei device.
2. i sistemi risultano sottoutilizzati durante tutto il loro utilizzo
3. i consumi dipendono da molti fattori tra il quale il pattern di utilizzo;

Futuri step

La tesi si è limitata all'analisi di un singolo cluster creato con `K3s`. Utilizzando il paradigma del `"computing liquido"` questo studio potrebbe essere espanso all' analisi delle potenzialita' di risparmio energetico ottenibili mediante la condivisione di risorse tra dispositivi IoT su `cluster federati` tramite l'utilizzo di **Liqo** [71]; una visione logica di tale concezione è visibile nella figura 6.1.

Liqo è un framework open-source sviluppato all'interno del "Computer Networks Group" del Politecnico di Torino.
Liqo consente a Kubernetes di condividere risorse e servizi in modo trasparente e

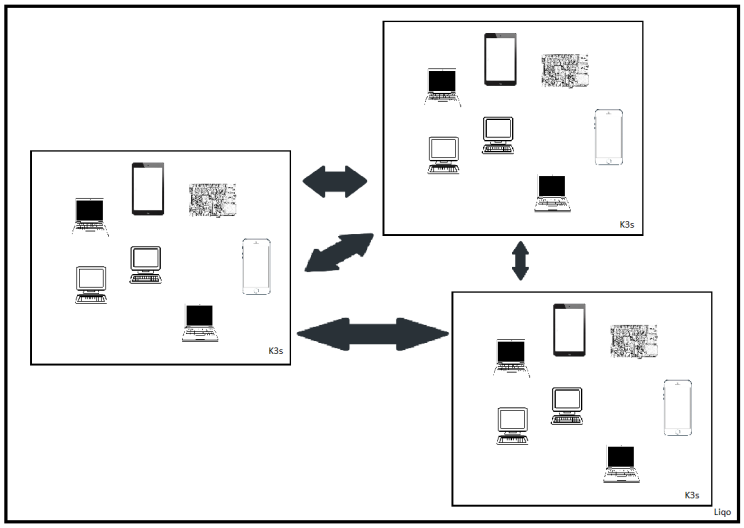


Figura 6.1: Una visione logica dei dispositivi con risorse condivise tra cluster diversi. I cluster condividono le risorse tra di loro grazie a `Liqo`.

sicuro, in modo che si possano eseguire task su qualsiasi altro cluster federato. Liqo supporta anche K3s, quindi anche singole macchine possono partecipare, creando data center dinamici e opportunistici che includono *computer desktop* e *laptop*. Grazie a Liqo quindi, i task di un cluster potrebbero essere eseguiti su un altro più potente, ottenendo un eventuale risparmio energetico.

Acronimi

AI Artificial Intelligence

API Application Programming Interface

APP Applicazione

CI Continuous Integration

CO_2 Anidride Carbonica

CPU Central Processing Unit

CRUD Create, Read, Update, Delete

CSV Comma-Separated Values

DB Database

DBMS Database Management System

DC Data Center

DPC Dynamic Power Consumption

DVFS Dynamic Voltage and Frequency Scaling

GPU Graphical Processing Unit

HA High Availability

HDD Hard Disk Drive

HVAC Heating, Ventilation and Air Conditioning

I/O Input/Output

ICT Information and Communications Technology

IE9 Internet Explorer 9

IoT Internet of Things

IP Internet Protocol

IT Information Technology

K8s Kubernates

MQTT Message Queue Telemetry Transport

NAS Network-Attached Storage

OS Operating System

PC Personal Computer

PDU Protocol Data Unit

PUE Power Usage Effectiveness

RAM Random-Access Memory

SBC Single-board computer

SPC Static Power Consumption

SPM Spindle Motor

SSD Solid-State Drive

TXT Trusted Execution Technology

UPS Uninterrupted Power Supply

VCM Voice Coil Motor

Bibliografia

[1] *Moore's Law*. URL: http://www.mooreslaw.org/ (cit. a p. 1).

[2] Jim Wright. *Buy servers only if you really need them*. Lug. 2004. URL: https://www.computerweekly.com/opinion/Buy-servers-only-if-you-really-need-them/ (cit. a p. 2).

[3] Luiz André Barroso e Urs Hölzle. «The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines». In: *Synthesis Lectures on Computer Architecture* 4.1 (2009), pp. 1–108. DOI: 10.2200/S00193ED1V01Y200905CAC006. eprint: https://doi.org/10.2200/S00193ED1V01Y200905CAC006. URL: https://doi.org/10.2200/S00193ED1V01Y200905CAC006 (cit. a p. 2).

[4] Randy Bias. *Elasticity is NOT #Cloud Computing ... Just Ask Google*. Nov. 2010. URL: http://cloudscaling.com/blog/cloud-computing/elasticity-is-not-cloud-computing-just-ask-google/ (cit. a p. 2).

[5] Michael Ogbole, Engr Ogbole e Ayomide Olagesin. «Cloud Systems and Applications : A Review». In: *International Journal of Scientific Research in Computer Science, Engineering and Information Technology* (feb. 2021), pp. 142–149. DOI: 10.32628/CSEIT217131 (cit. a p. 3).

[6] *Kubernetes project*. URL: https://kubernetes.io/ (cit. a p. 3).

[7] Mattia Lavacca. «Scheduling Jobs on Federation of Kubernetes Clusters». master. Politecnico di Torino, 2020. Cap. 2 (cit. a p. 6).

[8] *Kubernetes official documentation*. URL: https://kubernetes.io/docs/home/ (cit. alle pp. 6, 13, 15).

[9] Abhishek Verma, Luis Pedrosa, Madhukar R. Korupolu, David Oppenheimer, Eric Tune e John Wilkes. «Large-scale cluster management at Google with Borg». In: *Proceedings of the European Conference on Computer Systems (EuroSys)*. Bordeaux, France, 2015 (cit. a p. 6).

[10] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek e John Wilkes. «Omega: flexible, scalable schedulers for large compute clusters». In: *SIGOPS European Conference on Computer Systems (EuroSys)*. Prague, Czech Republic, 2013, pp. 351–364. URL: http://eurosys2013.tudos.org/wp-content/uploads/2013/paper/Schwarzkopf.pdf (cit. a p. 6).

[11] Ferenc Hámori. *The History of Kubernetes on a Timeline*. Giu. 2018. URL: https://blog.risingstack.com/the-history-of-kubernetes/ (cit. a p. 7).

[12] Steven J. Vaughan-Nichols. *The five reasons Kubernetes won the container orchestration wars*. Gen. 2019. URL: https://blogs.dxc.technology/2019/01/28/the-five-reasons-kubernetes-won-the-container-orchestration-wars/ (cit. a p. 7).

[13] Kalyan Ramanathan. *5 business reasons why every CIO should consider Kubernetes*. Ott. 2019. URL: https://www.sumologic.com/blog/why-use-kubernetes/ (cit. a p. 7).

[14] Eric Carter. *Sysdig 2019 Container Usage Report: New Kubernetes and security insights*. Ott. 2019. URL: https://sysdig.com/blog/sysdig-2019-container-usage-report/ (cit. a p. 9).

[15] Diego Ongaro e John Ousterhout. «In search of an understandable consensus algorithm». In: *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*. 2014, pp. 305–319 (cit. a p. 10).

[16] *Kubernetes API official documentation*. URL: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/ (cit. a p. 13).

[17] *Using a k3s Kubernetes Cluster for Your GitLab Project*. URL: https://betterprogramming.pub/using-a-k3s-kubernetes-cluster-for-your-gitlab-project-b0b035c291a9 (cit. a p. 16).

[18] *K3s project*. URL: https://k3s.io/ (cit. alle pp. 17, 20, 21).

[19] *K3s*. URL: https://community.suse.com/posts/introduction-to-k3s#:~:text=K3s%20has%20four%20deployment%20architectures,Let's%20see%20how%20those%20work.&text=The%20simplest%20way%20to%20deploy,with%20the%20embedded%20agent%20enabled. (cit. a p. 19).

[20] Ward Van Heddeghem, Sofie Lambert, Bart Lannoo, Didier Colle, Mario Pickavet e Piet Demeester. «Trends in worldwide ICT electricity consumption from 2007 to 2012». In: *Computer Communications* 50 (2014). Green Networking, pp. 64–76. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2014.02.008. URL: https://www.sciencedirect.com/science/article/pii/S0140366414000619 (cit. a p. 23).

[21] Auday Al-Dulaimy, Wassim Itani, Ahmed Zekri e Rached Zantout. «Power management in virtualized data centers: State of the art». In: *Journal of Cloud Computing: Advances, Systems and Applications.* 5 (apr. 2016). DOI: 10.1186/s13677-016-0055-y (cit. alle pp. 24, 26).

[22] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith e Jonathan Koomey. «Recalibrating global data center energy-use estimates». In: *Science* 367 (feb. 2020), pp. 984–986. DOI: 10.1126/science.aba3758 (cit. alle pp. 25–27).

[23] George Kamiya. «Data Centres and Data Transmission Networks». In: *IEA, Paris* (giu. 2020). URL: https://www.iea.org/reports/data-centres-and-data-transmission-networks (cit. alle pp. 27, 28).

[24] Lu Liu, Osama Masfary e Nick Antonopoulos. «Energy Performance Assessment of Virtualization Technologies Using Small Environmental Monitoring Sensors». In: *Sensors (Basel, Switzerland)* 12 (dic. 2012), pp. 6610–28. DOI: 10.3390/s120506610 (cit. a p. 29).

[25] Corey Gough, Ian Steiner e Winston A. Saunders. *Energy Efficient Servers: Blueprints for Data Center Optimization.* 1st. USA: Apress, 2015. ISBN: 1430266376 (cit. alle pp. 29, 51).

[26] M. Avgerinou, P. Bertoldi e L. Castellazzi. «Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency». In: *Energies* 10 (2017), pp. 1–18 (cit. a p. 30).

[27] *How Much Energy Do Data Centers Really Use?* URL: https://energyinnovation.org/2020/12/27/ (cit. a p. 31).

[28] Miyuru Dayarathna, Yonggang Wen e Rui Fan. «Data Center Energy Consumption Modeling: A Survey». In: *IEEE Communications Surveys Tutorials* 18.1 (2016), pp. 732–794. DOI: 10.1109/COMST.2015.2481183 (cit. alle pp. 32, 38–40, 42, 44–47).

[29] Miyuru Dayarathna, Yonggang Wen e Rui Fan. «Data Center Energy Consumption Modeling: A Survey». In: *IEEE Communications Surveys Tutorials* 18.1 (2016), pp. 732–794. DOI: 10.1109/COMST.2015.2481183 (cit. alle pp. 30, 33).

[30] Sparsh Mittal. «Power Management Techniques for Data Centers: A Survey». In: (apr. 2014) (cit. a p. 32).

[31] Urs Hölzle Luiz André Barroso Jimmy Clidaras. *The Datacenter as a Computer. An Introduction to the Design of Warehouse-Scale Machines Second Edition.* Morgan Claypool Publishers, 2013 (cit. alle pp. 34, 50).

[32] Lu Liu, Osama Masfary e Nick Antonopoulos. «Energy Performance Assessment of Virtualization Technologies Using Small Environmental Monitoring Sensors». In: *Sensors (Basel, Switzerland)* 12 (dic. 2012), pp. 6610–28. DOI: 10.3390/s120506610 (cit. alle pp. 35, 47, 48).

[33] Arka A. Bhattacharya, David Culler, Aman Kansal, Sriram Govindan e Sriram Sankar. «The need for speed and stability in data center power capping». In: (2012), pp. 1–10. DOI: 10.1109/IGCC.2012.6322253 (cit. alle pp. 35, 36).

[34] Nicola Jones. «How to stop data centres from gobbling up the world's electricity». In: *Nature* 561 (set. 2018), pp. 163–166. DOI: 10.1038/d41586-018-06610-y (cit. alle pp. 37, 38).

[35] Lukas G. Swan e V. Ismet Ugursal. «Modeling of end-use energy consumption in the residential sector: A review of modeling techniques». In: *Renewable and Sustainable Energy Reviews* 13.8 (2009), pp. 1819–1835. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2008.09.033. URL: https://www.sciencedirect.com/science/article/pii/S1364032108001949 (cit. a p. 37).

[36] Chaoqiang Jin, Xuelian Bai, Chao Yang, Wangxin Mao e Xin Xu. «A review of power consumption models of servers in data centers». In: *Applied Energy* 265 (2020), p. 114806. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2020.114806. URL: https://www.sciencedirect.com/science/article/pii/S0306261920303184 (cit. a p. 38).

[37] Krishna T. Malladi, Frank A. Nothaft, Karthika Periyathambi, Benjamin C. Lee, Christos Kozyrakis e Mark Horowitz. «Towards energy-proportional datacenter memory with mobile DRAM». In: (2012), pp. 37–48. DOI: 10.1109/ISCA.2012.6237004 (cit. a p. 42).

[38] «United States Data Center Energy Usage Report». In: *e Federal Energy Management Program of the U.S. Department of Energy under Lawrence Berkeley National Laboratory Contract* (giu. 2016) (cit. alle pp. 43, 44).

[39] Congfeng Jiang, Yumei Wang, Dongyang Ou, Youhuizi Li, Jilin Zhang, Jian Wan, Bing Luo e Weisong Shi. «Energy efficiency comparison of hypervisors». In: *Sustainable Computing: Informatics and Systems* 22 (2019), pp. 311–321. ISSN: 2210-5379. DOI: https://doi.org/10.1016/j.suscom.2017.09.005. URL: https://www.sciencedirect.com/science/article/pii/S2210537917300963 (cit. a p. 47).

[40] Chaoqiang Jin, Xuelian Bai, Chao Yang, Wangxin Mao e Xin Xu. «A review of power consumption models of servers in data centers». In: *Applied Energy* 265 (2020), p. 114806. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2020.114806. URL: https://www.sciencedirect.com/science/article/pii/S0306261920303184 (cit. a p. 49).

[41] Jóakim von Kistowski, Hansfried Block, John Beckett, Cloyce Spradling, Klaus-Dieter Lange e Samuel Kounev. «Variations in CPU Power Consumption». In: (mar. 2016). DOI: 10.1145/2851553.2851567 (cit. a p. 49).

[42] Constanţa Zoie Rădulescu e Delia Mihaela Rădulescu. «A performance and power consumption analysis based on processor power models». In: (2020), pp. 1–4. DOI: 10.1109/ECAI50035.2020.9223124 (cit. a p. 52).

[43] Xiaobo Fan, Wolf-Dietrich Weber e Luiz Andre Barroso. «Power Provisioning for a Warehouse-Sized Computer». In: *SIGARCH Comput. Archit. News* 35.2 (giu. 2007), pp. 13–23. ISSN: 0163-5964. DOI: 10.1145/1273440.1250665. URL: https://doi.org/10.1145/1273440.1250665 (cit. alle pp. 52, 53).

[44] *Computer penetration rate among households worldwide 2005-2019*. URL: https://www.statista.com/statistics/748551/worldwide-households-with-computer/ (cit. alle pp. 52, 61).

[45] Pal A. De' R Pandey N. «Impact of digital surge during Covid-19 pandemic: A viewpoint on research and practice.» In: (2020). DOI: 10.1016/j.ijinfomgt.2020.1021716 (cit. a p. 53).

[46] Pavel Somavat e Vinod Namboodiri. «Energy Consumption of Personal Computing Including Portable Communication Devices». In: *Journal of Green Engineering* 1 (lug. 2011) (cit. a p. 54).

[47] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon Blair e Adrian Friday. *The climate impact of ICT: A review of estimates, trends and regulations*. Feb. 2021 (cit. a p. 54).

[48] Vasily Moshnyaga. «How to Really Save Computer Energy?» In: gen. 2008, pp. 89–95 (cit. alle pp. 55, 58, 61).

[49] Anna Menezes, Noah Nkonge, Andrew Cripps, R.A. Buswell e Dino Bouchlaghem. «Review of benchmarks for small power consumption in office buildings». In: (gen. 2012) (cit. alle pp. 55, 58).

[50] Dipartimento di Energetica Politecnico di Milano. «Misure dei consumi di energia elettrica nel settore domestico; Risultati delle campagne di rilevamento dei consumi elettrici presso 110 abitazioni in Italia». In: (2004) (cit. a p. 55).

[51] *Sito Micene - Politecnico di Milano*. URL: https://www.eerg.it/index.php?p=Progetti_-_MICENE (cit. a p. 55).

[52] Fatih Issi e Orhan Kaplan. «The Determination of Load Profiles and Power Consumptions of Home Appliances». In: *Energies* 11.3 (2018). ISSN: 1996-1073. DOI: 10.3390/en11030607. URL: https://www.mdpi.com/1996-1073/11/3/607 (cit. a p. 55).

[53] Tao Li e Lizy John. «Run-time Modeling and Estimation of Operating System Power Consumption». In: *ACM SIGMETRICS Performance Evaluation Review* 31 (mag. 2003). DOI: 10.1145/781027.781048 (cit. a p. 56).

[54] Giuseppe Procaccianti, Luca Ardito, Antonio Vetro e Maurizio Morisio. «Energy Efficiency in the ICT - Profiling Power Consumption in Desktop Computer Systems». In: ott. 2012, pp. 353–372. ISBN: 978-953-51-0800-9. DOI: 10.5772/48237 (cit. alle pp. 56, 57).

[55] Megan Bray. «Review of Computer Energy Consumption and Potential Savings». In: (2006) (cit. alle pp. 57, 60).

[56] Aqeel Mahesri e Vibhore Vardhan. «Power Consumption Breakdown on a Modern Laptop». In: *Proceedings of the 4th International Conference on Power-Aware Computer Systems*. PACS'04. Portland, OR: Springer-Verlag, 2004, pp. 165–180. ISBN: 3540297901. DOI: 10.1007/11574859_12. URL: https://doi.org/10.1007/11574859_12 (cit. a p. 58).

[57] Aníbal de Almeida, Paula Fonseca, Barbara Schlomann, Nicolai Feilberg e C. Ferreira. «Residential monitoring to decrease energy use and carbon emissions in Europe». In: *Eedal'06* (gen. 2006), pp. 1–17 (cit. alle pp. 59, 60).

[58] Jinwei Wang, Yuzhong Sun e Jianping Fan. «Analysis on Resource Utilization Patterns of Office Computer.» In: gen. 2005, pp. 626–631 (cit. a p. 61).

[59] Hanna Pihkola, Mikko Hongisto, Olli Apilo e Mika Lasanen. «Evaluating the Energy Consumption of Mobile Data Transfer—From Technology Development to Consumer Behaviour and Life Cycle Thinking». In: *Sustainability* 10.7 (2018). ISSN: 2071-1050. DOI: 10.3390/su10072494. URL: https://www.mdpi.com/2071-1050/10/7/2494 (cit. a p. 61).

[60] *Sito ITU - International Telecommunication Union*. URL: https://www.itu.int (cit. alle pp. 61, 62).

[61] Marco Torchiano, Giuseppe Procaccianti, Luca Ardito e Giuseppe Migliore. «Profiling Power Consumption on Mobile Devices». In: mar. 2013, pp. 101–106 (cit. a p. 62).

[62] Roberto Morabito. «Virtualization on Internet of Things Edge Devices With Container Technologies: A Performance Evaluation». In: *IEEE Access* 5 (2017), pp. 8835–8850. DOI: 10.1109/ACCESS.2017.2704444 (cit. alle pp. 63, 64).

[63] Munam Shah e Waqas Anwaar. «Energy Efficient Computing: A Comparison of Raspberry PI with Modern Devices». In: *International Journal of Computer and Information Technology* 4 (gen. 2015), pp. 410–413 (cit. a p. 63).

[64] Fabian Kaup, Philip Gottschling e David Hausheer. «PowerPi: Measuring and modeling the power consumption of the Raspberry Pi». In: *39th Annual IEEE Conference on Local Computer Networks*. 2014, pp. 236–243. DOI: 10.1109/LCN.2014.6925777 (cit. alle pp. 63, 64).

[65] *Meross IoT*. URL: https://pypi.org/project/meross-iot (cit. alle pp. 66, 67, 69, 70).

[66] *Sito Meross*. URL: https://www.meross.com/ (cit. a p. 66).

[67] *Sito psutil*. URL: https://pypi.org/project/psutil/ (cit. a p. 72).

[68] Claudio Usai. «Delivering Resilient Virtualized Services in Smart Grid Environments». Master Thesis. Politecnico di Torino, 2021 (cit. a p. 84).

[69] *Sito sysstat*. URL: https://github.com/sysstat/sysstat (cit. a p. 84).

[70] *Sito cpubenchmark*. URL: https://www.cpubenchmark.net/ (cit. a p. 84).

[71] *Sito Liqo*. URL: https://liqo.io/ (cit. a p. 87).