UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

Gestione del consumo energetico nel kernel Linux

Laureando

Alessandro Claudio Damo

Matricola 1201694

Relatore

Prof. Mauro Migliardi

Università degli Studi di Padova

Anno accademico 2023/2024

Sommario

Negli ultimi tempi è sempre più crescente l'importanza della gestione dell'energia, e tale incombenza si è manifestata anche sui sistemi operativi, ormai punto attorno alla quale le nostre vite gravitano più o meno indirettamente.
Tale necessità è data dallo scopo di ridurre il più possibile l'impatto ambientale e dalla volontà di ridurre il più possibile il consumo elettrico dato il suo progressivo aumento di costo.
Per sopperire a questo, anche il kernel Linux ha implementato metodi per la riduzione del consumo energetico, cercando allo stesso tempo però di compromettere il meno possibile le prestazioni.
Questa tesi vuole quindi concentrarsi su quali siano e come funzionino tali strumenti di gestione dell'energia nel kernel: verranno presi come esempio e definiti argomenti come l'ACPI (Advanced Configuration and Power Interface), il DVFS (Dynamic Voltage and Frequency Scaling) e il meccanismo runtime power management.
In maniera più specifica e particolare si vedrà come il kernel Linux sfrutta l'ACPI per gestire i dispositivi hardware e i loro stati di alimentazione, permettendo il funzionamento di meccanismi quali la sospensione alla RAM e al disco.
Si parlerà anche del DVFS, tecnica che permette la regolazione dinamica della frequenza della CPU per trovare il miglior equilibrio prestazioni e consumo energetico attraverso le logiche dei governor.
Infine, viene spiegato il meccanismo del Runtime power management, che permette ai dispositivi inattivi e non temporaneamente utilizzati di passare in maniera automatica quando necessario a stati di basso consumo, contribuendo così a ridurre il consumo energetico globale del sistema.
Questa tesi cerca in sunto di offrire una panoramica sulla gestione energetica nel kernel Linux presentando le soluzioni attualmente implementate e definite precedentemente, cercando dapprima di definirle e poi analizzando il loro funzionamento attraverso riferimenti a funzioni, commenti e documentazione presenti nel codice sorgente del kernel Linux.

Indice

1 Introduzione

In questo capitolo andremo a vedere le motivazioni che hanno portato alla scelta di questo argomento di tesi, sia riguardante il concetto di risparmio energetico, tema cardine al giorno d'oggi, sia riguardante il sistema operativo scelto, ovvero il sistema operativo open source Linux, con un breve accenno storico sulla sua nascita e la sua diffusione nel tempo.

1.1 Importanza della gestione del consumo energetico e obbiettivi della tesi

Il concetto di energia negli ultimi anni è diventato un tema centrale: se prima era un tema di discussione per una ristretta platea scientifica, a oggi è diventato un tema importante che ha raggiunto anche il cittadino più comune ed è uno dei principali soggetti di dibattito politico.

Questo perché gli ancora attuali conflitti geopolitici, l'aumento dei costi delle materie prime e la sempre più crescente domanda di energia hanno portato a un innalzamento dei costi dell'energia, cosa che ha causato in maniera più e meno diretta il un innalzamento importante del costo della vita del cittadino medio.

Per contestualizzare l'importanza della conseguenza dell'innalzamento dei costi, l'impiego dell'energia è la più importante spesa operativa di un edificio per uffici commerciali negli Stati Uniti e inoltre rappresenta circa un terzo del tipico budget operativo di un'azienda.[1]

Inoltre tra le cause associate a un altro importante tema ovvero l'impatto ambientale la produzione di energia ne risulta come una delle principali cause, poiché una buona parte dell'energia consumata a oggi proviene da fonti di tipo fossile come petrolio, gas carbone, che sono a loro volta responsabile di rilasciare grandi quantità di CO_2 e altri gas serra.

[1] Cos'è la gestione dell'energia? | IBM, https://www.ibm.com/it-it/topics/energy-management, 2024

Insomma il mondo ha bisogno di energia, e ha altrettanto bisogno di risparmiare energia.

Nel mondo dell'informatica, storicamente il focus nel passato è sempre stato di dare come priorità nei dispositivi a capacità di elaborazione, prestazioni e velocità di esecuzione di operazioni sempre più complesse.

La gestione del consumo energetico è un concetto diventato importante con l'avvento di dispositivi portatili di grandissima diffusione come telefonini, tablet, laptop e sistemi di gioco portatili, dove l'autonomia della batteria diventa un elemento importante per superare e battere la concorrenza.

Negli ultimi anni il concetto di risparmio energetico oltre ai dispositivi mobile ha raggiunto anche le piattaforme fisse, con l'avvento e la crescente diffusione del cloud computing e del conseguente aumento dei server.

Tali richiedono un'enorme quantità di energia non solo per l'elaborazione e la computazione, ma anche per il loro raffreddamento: quindi diminuire il più possibile i consumi dei server porterebbe a una conseguente e importante diminuzione dei costi operativi.

A oggi, quindi nella progettazione e nella costruzione di qualsiasi sistema informatico moderno la gestione energetica non è più un'opzione, ma anche una necessità fondamentale.

Il sistema operativo può giocare un ruolo importante nella gestione dell'energia, e l'obbiettivo di questa tesi è proprio di mostrare come con l'uso di alcuni meccanismi già implementati il sistema operativo open source Linux riesce a oggi a risparmiare energia con il giusto compromesso di prestazioni.

Vedremo concetti come ACPI, uno standard industriale aperto reso disponibile presente dal 1996 e sviluppato da grosse aziende del mondo dell'hi-tech come HP, Intel, Microsoft, Phoenix e Toshiba, che permette la gestione energetica dei sistemi da parte del sistema operativo, e approfondiremo il funzionamento di due dei vari tipi di sospensione globale del sistema.

Vedremo il meccanismo di Dynamic Voltage and Frequency Scaling, spesso abbreviato con il suo acronimo DVFS, che permette una regolazione intelligente della frequenza e della tensione della CPU in modo da abbassare i consumi con per esempio carichi di lavoro bassi.

Infine vedremo il runtime power management, che permette la sospensione automatica di singoli dispositivi se al momento non utilizzati, e di poter a loro volta essere riattivati velocemente quando necessario.

Infine cercheremo di trarre delle conclusioni, paragonando come possibile i consumi con altri sistemi operativi per ragionare su possibili strade da percorrere per migliorare ancora il risparmio dei consumi energetici.

1.2 Kernel Linux: motivazioni, storia e diffusione del kernel Linux

Partiamo innanzitutto introducendo il sistema operativo che ci permetterà di esplorare i meccanismi di gestione energetica, ovvero Linux. Con la sua essenza open source Linux ci permette di esplorare a fondo i meccanismi per il risparmio energetico, senza approssimazioni o speculazioni sul funzionamento di alcuni meccanismi magari di proprietà e quindi a noi segreti.

Storicamente, la genesi del kernel Linux ha inizio negli anni 80, quando Richard Stallman, lavoratore al tempo del MIT, si trovò un giorno incapace di far funzionare correttamente una stampante: tale dispositivo infatti si inceppava continuamente e nessuno veniva avvisato. Siccome il software che gestiva la stampante era proprietario e non poteva essere quindi modificato, Stallman si trovò totalmente inabilitato nel sistemare il problema

Questo evento portò Stallman a riflettere sull'importanza di avere accesso libero al codice sorgente del software, per poterlo modificare, migliorare e nel suo caso sistemare.

Fu questa esperienza a portarlo a ideare e creare il progetto GNU nel 1983: tale progetto infatti aveva l'obiettivo di sviluppare un sistema operativo libero e gratuito che rispettasse la libertà degli utenti.

Nel corso del tempo, il Progetto GNU ha realizzato numerosi obiettivi significativi che hanno trasformato il panorama del software libero e open source: troviamo per esempio il GCC (GNU Compiler Collection) ovvero un compilatore libero e open source per diversi linguaggi di programmazione, la libreria C glibc e la shell di comando GNU Bash; un kernel di GNU ha tardato ad arrivare.

Dobbiamo infatti aspettare fino al 1991 con il lavoro di Linus Torvalds, uno studente finlandese che iniziò a sviluppare un kernel per un futuro sistema operativo che doveva essere simile a UNIX, di nome Linux.

Tale kernel venne rilasciato nella sua versione 0.01 più rudimentale il 17 settembre del 1991, funzionava solo su alcuni modelli di computer basati su processori Intel 386, e non era ancora in grado di avviare un vero sistema operativo da sola: infatti necessitava del sistema operativo MINIX per la sua configurazione e compilazione.

Nato inizialmente come un hobby personale e non come un sistema commerciale professionale, in quanto libero e open source il kernel Linux soddisfava perfettamente le condizioni richieste dal progetto GNU.

Il primo approccio tra GNU e Linux avvenne nel rilascio del kernel nella versione 0.02, che utilizzava il supporto di strumenti GNU come il GCC e la shell Bash già definiti in precedenza.

Il secondo approccio avvenne quando Torvalds decise di rilasciare il kernel sotto licenza GPL (General Public License) del progetto GNU. Questa scelta trasformò il progetto Linux da un semplice hobby a un progetto libero e open source, aprendo la strada a una vasta comunità di sviluppatori da tutto il mondo, che iniziarono a contribuire attivamente.

L'ormai inevitabile connubio avvenne con il rilascio della prima versione stabile di Linux 1.0 nel 14 marzo 1994 consolidando il kernel come parte integrante di un sistema libero adesso chiamato GNU/Linux[2].

Da allora, il kernel ha continuato a crescere in complessità e versatilità: dalle circa 176.000 linee di codice oggi contiene oltre 27 milioni di linee di codice.

La diffusione del kernel Linux nel tempo è stata straordinaria: è utilizzato su un'ampia varietà di dispositivi, dai supercomputer agli smartphone, fino ai sistemi embedded.

Ha trovato successo soprattutto in ambito server e data center: oggi infatti, oltre il 96% dei principali server web utilizza Linux[3], inclusi servizi come Google, Twitter e Amazon.

Più relativa e limitata la sua presenza nel mercato dei sistemi operativi desktop (circa il 4%)[4], Linux però domina anche nel mercato dei supercomputer e nei dispositivi embedded, compresi molti dispositivi IoT.

Sono state la comunità e l'infrastruttura di sviluppo aperto le caratteristiche per il continuo miglioramento del kernel. Con il contributo di migliaia di sviluppatori da tutto il mondo, il kernel Linux ha visto l'introduzione di importanti miglioramenti, come il supporto per architetture aziendali, inclusi i mainframe IBM che hanno portato a un costante aggiornamento di funzionalità.

---

[2] Tringali L., Storia di GNU Linux, *GNU Linux Magazine Italia*, https://www.gnuLinuxmagazine.it/2020/07/storia- di-gnu-Linux/, 28/7/20.

[3] Linux Statistics Statistics: Market Data Report 2024, https://worldmetrics.org/Linux-statistics/, 23/7/24.

[4] Desktop Operating System Market Share Worldwide | Statcounter Global Stats, https://gs.statcounter.com/os-market-share/desktop/worldwide/, 14/10/24.

2 ACPI e la gestione degli Stati di Alimentazione in Linux

In questo capitolo andremo a vedere lo standard ACPI (e la sua implementazione APICA) e il suo utilizzo da parte del sistema operativo per la gestione del consumo energetico. Definiremo gli stati di alimentazione in cui può entrare un sistema e i singoli device, e infine vedremo come funzionano e vengono implementate la sospensione e la ibernazione dei sistemi globali.

2.1 ACPI e ACPICA

ACPI (Advanced Configuration and Power Interface) è uno standard industriale sviluppato da Intel, Microsoft e Toshiba, con il supporto di altre aziende, che definisce interfacce per il sistema operativo per scoprire, configurare, gestire il consumo energetico e monitorare lo stato dei dispositivi hardware del computer.

ACPICA (ACPI Component Architecture) è un progetto open-source che fornisce un'implementazione dello standard ACPI (Advanced Configuration and Power Interface) citato poc'anzi, e viene hostata su qualsiasi sistema operativo (nel nostro caso Linux).

Proprio in riferimento all'open source, il codice sorgente di ACPICA ha una licenza doppia (dual-licensed), in modo che possa essere ospitato sia su Linux come nel nostro caso sia su FreeBSD.

Struttura ACPICA[5]

La struttura di ACPICA è composta da due componenti principali più altri componenti più secondari. (Fig 2.1)

[5] Le informazioni riguardante la struttura di ACPICA sono prese da "ACPI in Linux"
Brown L., Keshavamurthy A.,Li D.S., Moore R., Pallipadi V., Yu L., 2005, ACPI in Linux, *Proceedings of the Linux Symposium,* vol. 1, pp. 51-69

Il primo dei due componenti principali è l'ACPI Core Subsystem, che è la parte principale di ACPICA che fornisce i servizi fondamentali per il suo funzionamento attraverso l'implementazione di alcuni strumenti. Questa parte, l'ACPI Core Subsystem, è totalmente indipendente dal sistema operativo, e rimane costante e identica su ogni piattaforma.

Il secondo dei due componenti più importanti è l'OS Service Layer (OSL) è la parte di ACPICA che si occupa delle comunicazioni tra ACPICA e il sistema operativo che lo ospita, e per tale motivo l'OSL è specifico e varia da sistema operativo a sistema operativo.

Altri componenti più secondari dell'ACPI sono un'interfaccia utente per le funzionalità di gestione dell'alimentazione e configurazione, un componente di gestione dell'alimentazione e delle politiche energetiche chiamato OSPM, un componente di gestione della configurazione.

Troviamo inoltre dei driver di dispositivo relativi e specifici ad ACPI (ad esempio, driver per il controller integrato, SM-Bus, batteria intelligente e batteria con metodo di controllo).

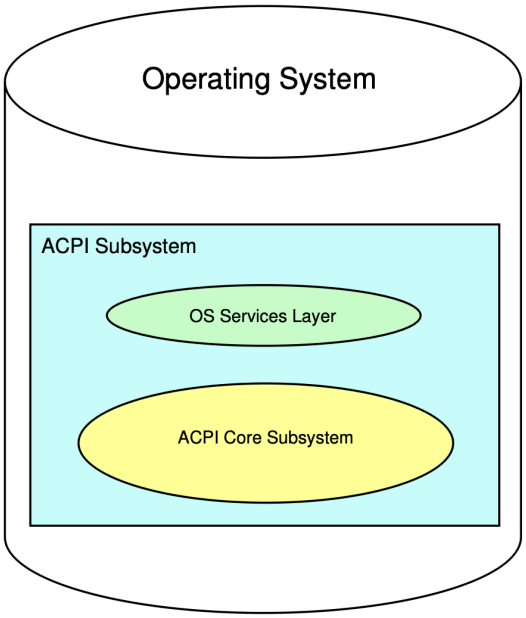

Fig 2.1 Architettura di ACPI
Fonte: ACPI in Linux, 2005

ACPI Core Subsystem

Abbiamo visto come ACPI Core Subsystem è composto da vari strumenti che permettono il funzionamento di ACPI, tra questi strumenti definiamo quelli principali:

Le tabelle ACPI sono diverse tabelle che contengono informazioni sull'hardware e sulle funzionalità del sistema: le più importanti tabelle che ci concede ACPI sono

principalmente la FADT e la DSDT, due tabelle che contengono informazioni importanti sull'hardware del sistema e informazioni sulle funzionalità di quest'ultime.

Per la gestione di tale tabelle è implementato anche un ACPI Table Management, mentre il lavoro di inizializzazione del contenuto delle tabelle viene compiuto dal Bios all'inizio della sessione.

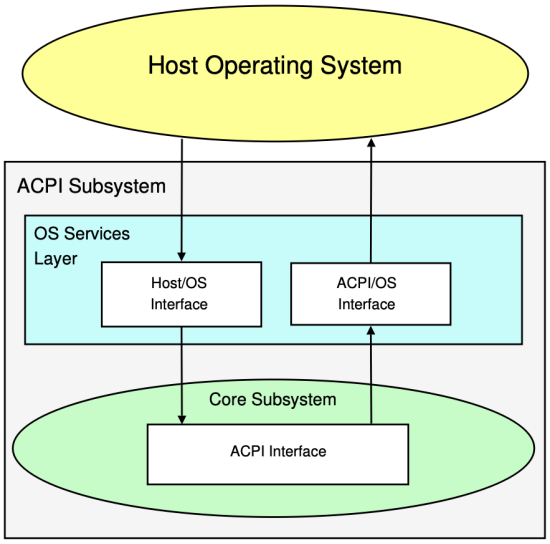

Fig 2.2 Interazione tra sistema operativo e ACPI core subsystem
Fonte: ACPI in Linux, 2005

Il Namespace ACPI è invece una struttura dati gerarchica che rappresenta una astrazione dell'hardware, in maniera da rendere più fruibile per i sistemi operativi un adattamento su una vasta gamma di configurazioni di hardware. La gestione di questa astrazione è compito del omonimo Namespace Management.

Entrambi questi strumenti sono scritti in un linguaggio binario specifico usato per descrivere le risorse hardware e le funzionalità di gestione dell'alimentazione chiamato Codice AML. Per interpretare ed eseguire questo esiste un apposito AML Interpreter.

OS service layer (OSL)

L'OS service layer si occupa di comunicare con il sistema operativo ospitante: Questo è possibile attraverso una traduzione delle richieste per i servizi ACPI provenienti dal sistema operativo host (e ovviamente dalle sue applicazioni) in chiamate al componente del Core Subsystem attraverso l'interfaccia Host/OS Interface.

Questa non è necessariamente una mappatura uno a uno: molto spesso invece, una singola richiesta del sistema operativo può essere tradotta in molte chiamate al Core Subsystem di ACPI.

Allo stesso modo viceversa l'ACPI/OS Interface traduce alcune funzioni del sistema operativo utili per alcune funzioni di ACPI core subsystem, permettendo una comunicazione tra OS e ACPI utile per alcuni componenti di ACPICA che necessitano di chiamate o risorse del sistema operativo.
(Fig 2.2)

2.2 Stati di alimentazione

La specifica ACPI definisce una serie di stati che regolano il comportamento delle principali componenti di un computer in relazione al risparmio energetico desiderato. In questo capitolo ci addentreremo nella definizione di quest'ultimi.[6]

Introduzione agli stati

Tali stati di alimentazione definiti da ACPI si organizzano in diverse categorie, come:

Stati globali (Gx) che indicano lo stato generale del sistema

Stati di sonno (Sx) che rappresentano gli stati di sospensione del sistema

Stati del processore (Cx) che si riferiscono agli stati di alimentazione del singolo processore

Stati dei dispositivi (Dx) che indicano lo stato di alimentazione dei singoli dispositivi hardware collegati.

Stati di performance (Px) che indicano gli stati di prestazione dei processori

Tali categorie di stati sono in relazione tra di loro, come vedremo successivamente.
(Fig 2.3)

---

[6] le informazioni di questo sezione sono prese dall'articolo "Gestione dell'alimentazione, parte 1: definizione degli stati ACPI"
Bassignana F., 2017, Gestione dell'alimentazione, parte 1: definizione degli stati ACPI - WEEE Open, https:// weeeopen.polito.it/gestione-dellalimentazione-parte-1-definizione-degli-stati-acpi/

I numeri invece indicano il livello di attività della categoria:
Gli stati 0 per esempio (come G0, S0, D0, ecc.) indicano tendenzialmente un sistema attivo e completamente operativo, pronto per l'interazione con l'utente; al contrario, gli stati con numeri più alti rappresentano condizioni di "sospensione" o "riposo", dove un numero maggiore corrisponde a consumi energetici inferiori, ma che richiedono più tempo per tornare allo stato attivo, fin ad arrivare a un numero massimo che rappresenta la piena attività.

Stati globali (Gx)

Gli stati globali descrivono lo stato dell'intero sistema e sono suddivisi in cinque principali categorie:

Working State (G0) dove lo stato attivo in cui il sistema è pienamente operativo. Tutti i componenti hardware funzionano e il sistema può eseguire qualsiasi operazione. In questo stato, il consumo energetico è massimo, poiché tutte le risorse sono attive. Il working state supporta e comprende soltanto lo stato di sonno S0, che indica uno stato di non sonno e piena attività.

Sleeping State (G1) dove lo stato che si riferisce a un insieme di stati di sospensione o sonno. La CPU smette di funzionare, ma alcune periferiche o moduli di sistema rimangono in uno stato pronto all'uso. Lo stato di sleeping state G1 comprende tutte le modalità di sonno (S1,S2,S3,S4).

Soft Off (G2) dove in questo stato il sistema è spento e non svolge alcuna attività operativa. Tuttavia, l'alimentazione è ancora presente per alcune funzionalità di base, come il riavvio tramite un segnale esterno (Wake-on-LAN, ad esempio). Comprende unicamente lo stato di sonno S5.

Mechanical Off (G3) che è lo stato completamente spento. Nessuno dei componenti è alimentato e ovviamente nessuna operazione può essere eseguita. Per riavviare il sistema è necessario un avvio totalmente manuale e tramite pulsante di accensione. Non comprende nessun stato di sonno poiché il sistema è completamente spento.

Stati di sonno (Sx)

Sono stati che descrivono cosa accade a livello di sistema, e sono suddivisi in cinque categorie:

S0: come abbiamo visto nello stato G0 lo stato operativo in cui il sistema è completamente attivo e funzionante.

S1: La CPU smette di eseguire istruzioni, ma il contesto della CPU (registro, cache, ecc.) rimane intatto. Viene risparmiata una quantità limitata di energia.

S2: In questo stato, la CPU è completamente disattivata, ma altre parti del sistema, come la RAM, rimangono alimentate.

S3: Conosciuto come suspend to RAM, questo stato spegne quasi tutto l'hardware tranne la memoria volatile. Il contesto del sistema viene memorizzato nella RAM, permettendo un rapido ritorno allo stato operativo. È uno degli stati di sospensione più utilizzati nei laptop.

S4: Detto anche hibernate o suspend to Disk, è uno stato in cui il contesto del sistema viene salvato su disco (memoria non volatile). In pratica, il sistema si spegne completamente, ma al successivo riavvio sarà possibile riprendere l'attività esattamente da dove era stata interrotta. Questo stato consuma ancora meno energia rispetto a S3

S5: come abbiamo visto con lo stato globale G2, in questo stato, il computer è spento e non esegue alcuna operazione, ma rimane ancora alimentato per alcune funzioni minime.

Stati del processore (Cx)

Gli stati Cx si applicano specificamente al processore e descrivono i vari livelli di risparmio energetico che può adottare quando non è necessario eseguire attività intensive.

Esistono diversi stati C, da C0 a Cn, con livelli crescenti di risparmio energetico:

C0: Lo stato attivo. Il processore esegue istruzioni e opera normalmente. In questo stato, il consumo energetico è massimo.

C1: Lo stato di Halt. In questo stato, il processore non esegue istruzioni, ma può riprendere immediatamente l'attività senza alcun ritardo significativo. È uno stato di basso consumo energetico, ma senza spegnere effettivamente nessuna parte del processore.

C2: Detto anche Stop-Clock, il processore in questo stato disattiva parti della sua logica interna per ridurre ulteriormente il consumo energetico. Il ritorno allo stato C0 richiede un breve ritardo.

C3: Conosciuto come Sleep State, in questo stato il processore interrompe completamente l'attività interna, ad eccezione di alcuni segnali di monitoraggio esterni necessari per riattivarlo. A differenza dello stato C2, per un ritorno a uno stato attivo C0 ci vorrà più tempo, ma i consumi in questo stato sono minori.

A seconda del design del processore, potrebbero esistere ulteriori stati (come C4 o C6) che riducono ulteriormente il consumo energetico.

Stati dei dispositivi (Dx)

Anche i dispositivi hardware come schede di rete, dischi rigidi e altri componenti periferici sono costruiti per supportare una serie di stati di risparmio Dx, che si applicano ai dispositivi hardware collegati al sistema.
Anche qui, gli stati seguono una numerazione simile, con D0 che rappresenta lo stato attivo e i numeri successivi che indicano crescenti livelli di sospensione e riduzione dell’energia. Abbiamo come stati definiti come standard:

D0: Stato operativo. Il dispositivo è completamente attivo e funziona normalmente.

D1: Stato di basso consumo energetico con alcune funzionalità ancora attive. Non è ampiamente utilizzato.

D2: Stato con risparmio energetico maggiore rispetto a D1, ma con tempi di ripristino più lunghi.

D3: Stato in cui il dispositivo è completamente spento. Il ripristino allo stato D0 richiede tempo e può richiedere un riavvio completo del dispositivo.

Stati di prestazione (Px)

Come abbiamo già visto, questi stati indicano i vari stati di prestazione dei processori quando il processore è attivo ovvero in stato C0.

Il numero di stati P (P0, P1, P2, etc…) è definito dal dispositivo o dalla CPU e non può superare 255.

Come vedremo nella parte riguardante il Dynamic Frequency Scaling, servono per aumentare ulteriormente il risparmio energetico mandando il processore su frequenze più basse o intermedie.

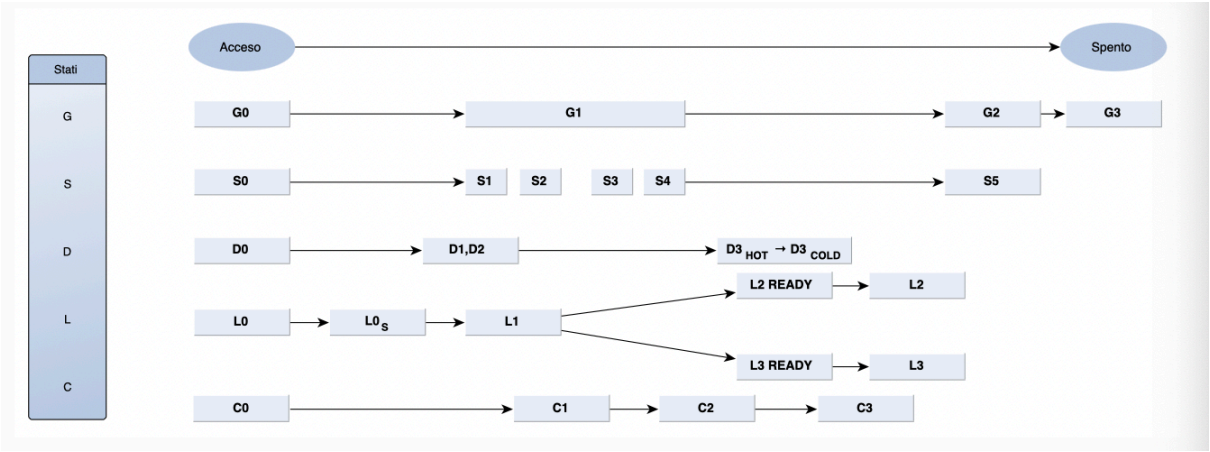

Fig 2.3 Stati di alimentazione di ACPI

Fonte: Gestione dell'alimentazione, parte 1: definizione degli stati ACPI, 2017

2.3 Meccanismi di Sospensione in Linux

La sospensione è uno stato di risparmio energetico in cui il sistema viene appunto sospeso, ovvero messo in uno grado di disattivazione che varia in base al tipo di sospensione scelta, ma in grado di riprendere dallo stesso esatto punto in cui sono state disattivate, con tempi diversi.
La sospensione può essere alla RAM (Suspend to RAM o sospensione) o al disco (Suspend to disk o ibernazione), o to idle. Quest'ultima non verrà approfondita in questo documento poiché poco utilizzata su Linux per via di problemi di ancora presenti di battery drain[7].

Sospensione to RAM (Suspend to RAM)

Durante la sospensione alla RAM (che è definito come stato di sonno ACPI S3), il sistema entra in uno stato di basso consumo, dove solo la RAM rimane alimentata per mantenere i dati. Il resto del sistema e dei dispositivi come il processore, schermo, dischi vengono spenti o messi in stato di inattività. Anche se il consumo energetico è basso, non è pari a zero.
Il vantaggio principale rispetto alla ibernazione (che vedremo successivamente) è la ripresa quasi immediata. Quando si riattiva il sistema, esso riprende rapidamente dallo stato salvato in RAM senza bisogno di riavviare completamente.
Questa modalità è ideale per pause brevi, come chiudere il laptop temporaneamente.

Sospensione to Disk (Ibernazione)

Nell'ibernazione (che è definito come stato di sonno ACPI S4), il contenuto della RAM viene copiato sul disco e dopodiché il sistema si spegne completamente. Questo significa che il consumo energetico è praticamente nullo mentre il computer è spento. Rispetto alla sospensione to RAM, l'ibernazione richiede più tempo per riprendere, perché i dati devono essere ricaricati dal disco alla RAM. Tuttavia a vantaggio rispetto

[7] Power management/Suspend and hibernate - ArchWiki, https://wiki.archLinux.org/title/Power_management/Suspend_and_hibernate, 2024

alla sospensione to RAM non c'è rischio di perdere dati in caso di interruzione dell'alimentazione, poiché tutto è stato salvato su disco, e con consumi decisamente minori.
Questa modalità è più adatta a interruzioni prolungate, come spegnere il computer durante la notte o durante un viaggio.

2.4 Implementazione della sospensione alla RAM in Linux

Vediamo la transizione da un sistema dallo stato G0 (Attivo) a uno stato di G1 di sonno che comprende gli stati (S3, and S4) come definito dalla ACPI Specification 6.5[8].
A dovere di informazione nel processo che descriveremo successivamente sono stati esclusi rispetto alla fonte i procedimenti per gli hardware reduced e per gli stati di sospensione S1 e S2.

1) Per prima cosa, il sistema operativo (in particolare il sottosistema che si occupa del power management) decide secondo le sue politiche di gestione energetica di mettere il sistema in uno stato di sonno, per esempio dopo un evento come la chiusura del coperchio di un laptop.
Nel caso di Linux, è il Power Management core (anche detto PM core) presente nella directory *kernel/power/* ad avviare il processo di sospensione attraverso la funzione visibile all'esterno chiamata *pm_suspend(suspend_state_t state)*[9] dove l'argomento passato è lo stato di sospensione (che può essere S3 o S4): tale funzione ha la responsabilità di chiamare tutte le funzioni responsabili dei passi successivi, come vediamo dalla immagine (fig 2.4)

[8] Il processo di sospensione seguito è quello descritto dalla documentazione di "ACPI Specification 6.5"
ACPI Specification 6.5, https://uefi.org/specs/ACPI/6.5/16_Waking_and_Sleeping.html, 2024

[9] Linux kernel 6.12, 2024, File 'suspend.c', https://github.com/torvalds/Linux

2) Il sistema operativo invoca il metodo _TTS di ACPI per indicare lo stato di sospensione più profondo a cui il sistema passerà che può essere sempre S3 o S4. Per il sistema operativo Linux la funzione che si occupa di eseguire questo metodo della funzione ACPI è *acpi_sleep_tts_switch(u32 acpi_state)*[10].

3) Il sistema operativo poi esamina i dispositivi abilitati per il risveglio e determina lo stato di sospensione più profondo per i singoli device supportati e compatibili con la sospensione scelta.

4) Il sistema operativo imposta tutti i driver dei dispositivi nel rispettivo stato Dx, precedentemente esaminato: i dispositivi abilitati al risveglio entrano nello stato Dx sufficiente per permettere un evento risveglio, mentre quelli non abilitati passano allo stato D3.
La funzione in Linux responsabile della messa in sospensione dei vari dispositivi al giusto stato è *dpm_suspend()*[11], che esegue le callback di sospensione dei driver di tutti i singoli dispositivi.

5) Il sistema operativo esegue il metodo _PTS (Prepare To Sleep) di ACPI, anche qui passando un argomento che indica lo stato di sospensione desiderato, come S3 o S4. La funzione che si occupa eseguire tale metodo per ACPI è *acpi_hw_execute_sleep_method("_PTS", acpi_state)*[12] che esegue il metodo _PTS di ACPI relativo alla sospensione.

---

[10] Linux kernel 6.12, 2024, File ‘sleep.c', https://github.com/torvalds/Linux

[11] Linux kernel 6.12, 2024, File ‘main.c', https://github.com/torvalds/Linux

[12] Linux kernel 6.12, 2024, File ‘hwesleep.c', https://github.com/torvalds/Linux

6) Il sistema operativo prima salva in memoria il contesto e poi disabilita i processori “secondari”, che sono i processori non responsabili dell'attuale processo di sospensione. Rimane attivo solo il processore “principale" che è quello che protrae e termina il processo di sospensione.

Il salvataggio del contesto è fondamentale in modo che al momento della ripresa il processore venga ripristinato in maniera identica al momento in cui è stato sospeso.
Il responsabile di tale operazione in Linux è la funzione
pm_sleep_disable_secondary_cpus()[13]

7) Il sistema operativo scrive il waking vector nella tabella di ACPI FACS (Firmware ACPI Control Structure).
Il waking vector è uno spazio di memoria responsabile di salvare l'indirizzo di memoria del codice che veniva eseguito subito prima dell'entrata in sospensione del sistema, così il sistema può riprendere esattamente dove aveva terminato.[14]
La funzione in Linux che si occupa di questa funzione è
acpi_hw_set_firmware_waking_vector()[15]

8) Il sistema operativo cancella il bit WAK_STS nei registri PM1a_STS e PM1b_STS. Questi sono registri di ACPI utilizzati per la ripresa del sistema, dove l'ultimo bit rappresenta il bit di WAK_STS, un bit che se impostato porterà il sistema a una transizione a uno stato di lavoro[16]. È fondamentale spegnere questo bit per evitare un erronea ripresa istantanea una volta conclusa la procedura di sospensione.

---

[13] Linux kernel 6.12, 2024, File 'power.h', https://github.com/torvalds/Linux

[14] ACPI Specification 6.5, https://uefi.org/specs/ACPI/6.5/16_Waking_and_Sleeping.html, 2024

[15] Linux kernel 6.12, 2024, File ‘hwxfsleep.c', https://github.com/torvalds/Linux

[16] ACPI Specification 6.4, https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/04_ACPI_Hardware_Specification/ACPI_Hardware_Specification.html, 2024

La funzione del kernel di Linux che si occupa di questa operazione è *acpi_hw_legacy_sleep()*[17], in particolare la sezione che chiama la funzione *acpi_write_bit_register(ACPI_BITREG_WAKE_STATUS, ACPI_CLEAR_STATUS);*

9) Il sistema operativo salva il contesto del processore principale rimanente in memoria, concludendo il salvataggio dei contesti in memoria di tutti i processori.

10) Il sistema operativo svuota le cache

11) Il sistema operativo abilita i segnali di risveglio tramite i registri responsabili della gestione dei General Purpose Events.
Questi eventi permettono il risveglio del sistema dallo stato di sospensione, come può essere per esempio la pressione di un tasto della tastiera[18]. La funzione che si occupa di questo lavoro è la funzione *acpi_enable_all_wakeup_gpes()*[19] che si occupa inoltre di disabilitare tutti gli altri eventi.

12) Il sistema operativo scrive nei registri PM1a_CNT e PM1b_CNT, registri dell'ACPI per la gestione della sospensione, scrivendo i valori sulle regioni del registro SLP_TYPb e SLP_EN.
Nella regione SLP_TYPb (grande 3 bit) di entrambi i registri viene scritto il tipo di sospensione (S3 o S4) e viene attivato il bit di SLP_TYPb dello stesso registro, procedendo alla terminazione del processo di sospensione scelto (S3 o S4) e scritto nei bit di SLP_TYPb.[20]

---

[17] Linux kernel 6.12, 2024, File 'hwsleep.c', https://github.com/torvalds/Linux

[18] ACPI Specification 6.4, https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/04_ACPI_Hardware_Specification/ACPI_Hardware_Specification.html, 2024

[19] Linux kernel 6.12, 2024, File 'evxfgpe.c', https://github.com/torvalds/Linux

[20] ACPI Specification 6.4, https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/04_ACPI_Hardware_Specification/ACPI_Hardware_Specification.html, 2024

Tutto questo avviene nella funzione *acpi_hw_legacy_sleep()*[21]

13) Il sistema è entrato finalmente nello stato di sospensione specificato.

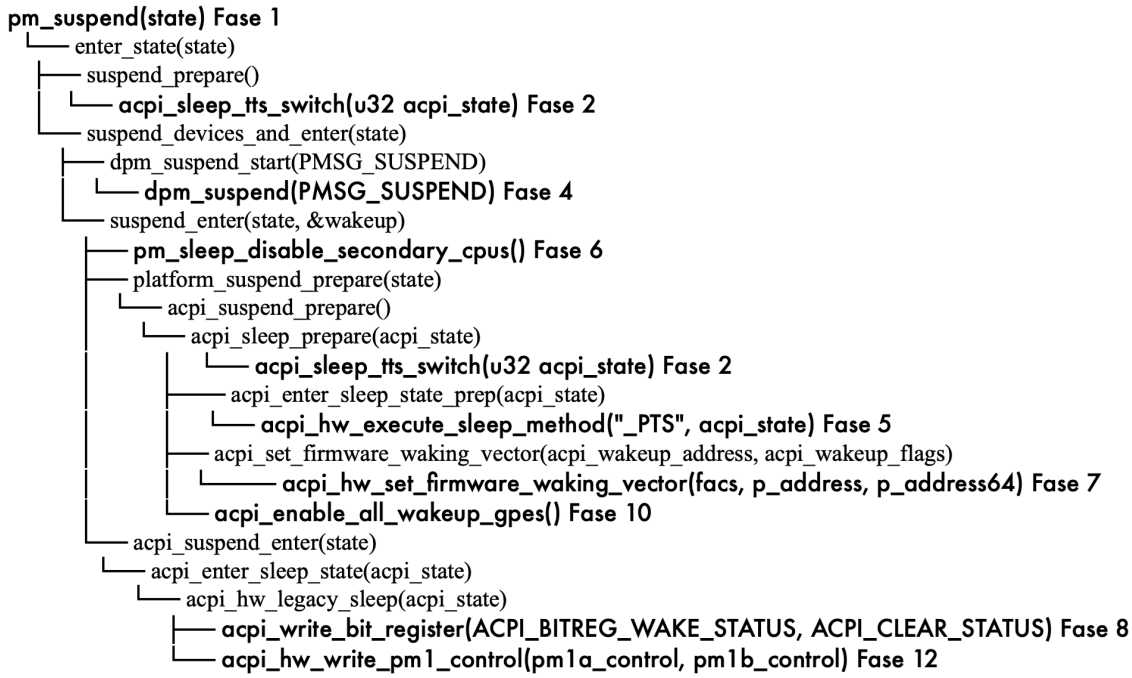

Fig 2.4 Riassunto del processo di sospensione di ACPI a partire dalla chiamata pm_suspend()

[21] Linux kernel 6.12, 2024, File 'hwsleep.c', https://github.com/torvalds/Linux

3 DVFS e la regolazione dinamica della frequenza del processore

In questo capitolo andremo a vedere il meccanismo di DVFS, e come i sistemi operativi attraverso la gestione dinamica della frequenza del processore riescono a risparmiare energia elettrica. Vedremo come attraverso i governor vengono implementate logiche e politiche per una regolazione intelligente della frequenza, e infine una implementazione del funzionamento del calcolo della frequenza e della sua applicazione.

3.1 DVFS

Il Dynamic Voltage and Frequency Scaling (DVFS) è una tecnica di gestione energetica utilizzata nei sistemi di calcolo per cercare un bilanciamento tra la prestazione e il consumo energetico.

Più specificamente è un meccanismo che consente di regolare dinamicamente la tensione e la frequenza del clock di un processore: Infatti sia la tensione che la frequenza del clock di un processore sono parametri che determinano sia la velocità con cui il processore esegue le operazioni sia il consumo energetico.

La frequenza di clock della CPU è un parametro che determina la velocità con cui vengono eseguite le istruzioni: a frequenze più alte, la CPU può elaborare più istruzioni per unità di tempo, ma il consumo energetico aumenta.

La tensione di alimentazione della CPU invece è un parametro direttamente collegato alla stabilità del funzionamento: a frequenze più alte, la CPU richiede tensioni maggiori per operare correttamente. Questo comporta che una tensione più alta ovviamente comporta sia un aumento del consumo di energia diretta sia indirettamente attraverso l'energia ulteriore necessaria per raffreddare il sistema ulteriormente.

Intuitivamente quindi viene naturale in caso di periodi di basso carico di lavoro cercare di ridurre la frequenza e la tensione per risparmiare energia.
Una approssimazione della potenza consumata da un processore moderno che conferma la nostra intuizione la possiamo avere e calcolare grazie alla equazione della potenza dinamica per circuiti CMOS (Complementary Metal-Oxide-Semiconductor) che ci conferma matematicamente che la relazione tra frequenza di clock, tensione e potenza consumata segue una legge quasi quadratica: e che quindi riducendo quindi la tensione e la frequenza, si può ottenere una riduzione significativa del consumo energetico.
Matematicamente, il consumo energetico della CPU può essere approssimato dalla formula[22]: $P = C \times V^2 \times f$

Dove:

P è la potenza consumata,

C è una costante legata alla capacità del processore,

V è la tensione di alimentazione della CPU,

f è la frequenza operativa della CPU.

Ricapitolando, riducendo la tensione e la frequenza, è possibile abbassare il consumo energetico totale della CPU, a discapito però delle prestazioni dovute dall'abbassamento della frequenza. Di conseguenza, il DVFS cerca di bilanciare prestazioni ed efficienza energetica e trovare un buon compromesso in base alle esigenze operative del sistema.

Il DVFS regola infatti questi due parametri in modo dinamico, cercando di trovare un bilanciamento: se il carico di lavoro è basso, il sistema operativo, in collaborazione con il processore, può abbassare la frequenza e la tensione della CPU per ridurre il consumo energetico. Quando il carico aumenta, il DVFS eleva la frequenza e la tensione per soddisfare la domanda di calcolo.

---

[22] Dynamic Power dissipation in CMOS - VLSI UNIVERSE, https://www.vlsiuniverse.com/dynamic-power-dissipation-in-cmos, 21/7/21.

3.2 Implementazione del DVFS in Linux attraverso CPUFREQ

Linux implementa la tecnica del DVFS attraverso CPUFreq, un sottosistema di Linux che consente la regolazione dinamica della frequenza della CPU.
Come abbiamo visto nel capitolo precedente è intuitivo e logico abbassare la frequenza della CPU quando non è necessario un carico di lavoro intenso, e ciò ci consente ovviamente di risparmiare energia. Nel caso di grossi carichi di lavoro, invece, sarebbe ideale poter riportare al massimo della frequenza il processore per fare in modo di non rallentare e intasare tutto il sistema. Per permettere questa regolazione dinamica e intelligente, CPUFreq si compone di vari elementi: Driver, governor e CoreCPUFreq.
Per quanto riguarda invece la tensione del processore invece invece, tali regolazione sono gestite dalla CPU stessa attraverso o il firmware o dei controller fisici di alimentazione a livello hardware direttamente integrati, valore di regolazione che dipende dalla frequenza scelta, in maniera da garantire la stabilità e il funzionamento della CPU stessa.

Driver di CPUFreq[23]

Sono Driver specifici della CPU che fanno da interfaccia tra CPUFreq e il processore. In Linux sono documentati i driver come intel_pstate che gestiscono il controllo della frequenza per le CPU Intel e i driver acpi-cpufreq che sono driver più generici per la gestione della frequenza attraverso ACPI sui valori di PState.

Core CPUFreq[24]

Core CPUFreq invece è la parte di CPUFrq che si occupa prevalentemente di gestire lo scaling della CPU, con implementazione di funzioni che permettono di gestire la

[23] CPUfreq - Red Hat Enterprise Linux 7 Power Management Guide, https://docs.redhat.com/en/documentation/red_hat_enterprise_Linux/7/html/power_management_guide/cpufreq_governors#cpufreq_drivers, 2024

[24] Governors, https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, 2024

registrazione dei driver, il monitoraggio e la gestione delle politiche di governor che vedremo successivamente.[25]

Governor di CPUFreq

I governor sono moduli di Linux che implementano politiche e logiche per il calcolo della frequenza del processore. In pratica sono i responsabili di calcolare una frequenza in base a delle logiche o dei feedback da parte di altri componenti del sistema operativo. Li vedremo in maniera più approfondita nel prossimo capitolo.

3.3 Governor implementati in Linux

Il governor come abbiamo anticipato poc'anzi è un componente software che gestisce e calcola la frequenza di clock della CPU, regolando staticamente o dinamicamente le prestazioni in base al carico di lavoro, in base alle esigenze energetiche e in base alle politiche dei governor stessi.

Esistono infatti più tipi di governor, che implementano politiche di scelta della frequenza cercando di coprire più filosofie di calcolo. In seguito troveremo infatti diversi esempi di governor presenti e implementati attualmente in Linux, definite nella sua documentazione.[26]

Performance

Il governor Performance di CPUFreq imposta la CPU in modo statico alla frequenza più alta possibile, entro i limiti definiti dalla policy.

---

[25] Il codice del core CPUfreq si trova raggruppato all'interno del file `driver/cpufreq/cpufreq.c` all'interno del codice sorgente.

[26] I governor descritti in questa sezione sono quelli descritti nella sezione di documentazione di Linux "Governors"
Governors, https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, 2024

Powersave

Il governor Powersave di CPUfreq imposta la CPU in modo statico alla frequenza più bassa possibile, entro i limiti definiti dalla policy.

Userspace

Il governor Userspace di CPUfreq permette all'utente, o a qualsiasi programma in spazio utente eseguito con UID "root", di impostare la CPU a una frequenza specifica tramite un file sysfs chiamato scaling_setspeed, disponibile nella directory del dispositivo CPU.

Ondemand

Il governor Ondemand modifica la frequenza del processore in base al carico di lavoro: quando il carico aumenta allora la frequenza cercherà di aumentare conseguentemente, allo stesso quando il carico diminuisce, la frequenza viene ridotta gradualmente, con un ritardo configurabile.
Il carico di lavoro viene calcolato in maniera approssimativa in base a quanto tempo la cpu in percentuale ha lavorato effettivamente e non è stata in idle dato un certo intervallo di tempo.
Questo governor era impostato come standard nelle vecchie versioni di Linux, attualmente invece è stato sostituito come predefinito da Schedutil.

Conservative

Il governor Conservative è simile a Ondemand ma con un approccio più cauto nell'aumentare la frequenza della CPU.
A differenza di quest'ultimo, che aumenta subito la frequenza al massimo quando rileva un carico di lavoro, conservative la incrementa gradualmente.
Allo stesso modo, la frequenza viene ridotta lentamente quando il carico diminuisce.

Regola la frequenza della CPU in base alle informazioni provenienti dallo scheduler del kernel, che è responsabile della distribuzione dei processi ai core della CPU.
Quando lo scheduler nota un aumento del carico di lavoro, il governor aumenta la frequenza della CPU per rispondere alla domanda; quando il carico diminuisce, la frequenza viene ridotta per risparmiare energia
Attualmente è il governor impostato in maniera predefinita in Linux attualmente (sostituendo il prima predefinito Ondemand) e per questo lo vedremo approfondito nel capitolo successivo.

3.4 Schedutil

Il governor Schedutil è una delle componenti chiave del sistema di gestione energetica del kernel Linux. Introdotto nel 2016 con il kernel 4.7 e sviluppato principalmente da Rafael J. Wysocki insieme ad altri collaboratori di Intel, è stato progettato per ottimizzare il bilanciamento tra prestazioni e consumo energetico.[27]

Il governor Schedutil rispetto al predecessore Ondemand utilizza una maniera alternativa e più immediata per cercare un bilancio tra prestazioni e consumi:
mentre come abbiamo già visto Ondemand utilizza informazioni direttamente dal processore (ovvero la percentuale di utilizzo o di idle del processore in un dato momento) il governor Schedutil lavora a stretto contatto con lo scheduler come il nome lascia intuire del kernel per determinare la frequenza ottimale della CPU, facendo in modo che la potenza di elaborazione sia adeguata alle esigenze del momento, riducendo al minimo il consumo energetico.

Il governor Schedutil è progettato per ricalcolare la frequenza ogni volta che uno o più processori nel sistema sperimentano cambiamenti nel carico di lavoro.

[27] Patch submission for the Linux ACPI project, https://patchwork.kernel.org/project/Linux-acpi/patch/4627718.FT18d2LR5p@vostro.rjw.lan/, 2024

Per esempio quando un task viene migrato a un processore con prestazioni superiori o quando il tasso di istruzioni desiderato aggregato di un processore si riduce, Schedutil garantisce che il processore operi a una frequenza inferiore; altrimenti, se avviene l'opposto, il processore funziona a una frequenza più alta.[28]

Logica e politica di Schedutil

Sempre nella documentazione di Linux[29], sono definite le logiche per il calcolo della frequenza ideale secondo Schedutil. Per fare questo sono necessarie due operazioni, una di monitoraggio e di stima del carico di lavoro e un'altra di calcolo della frequenza data la stima di carico di lavoro precedentemente stimata.

Monitoraggio del carico di lavoro

Per monitorare il carico di lavoro sui processori affinché si abbia un calcolo coerente per la nuova frequenza, il governor Schedutil utilizza l'algoritmo PELT (Per Entity Load Tracking).

L'algoritmo PELT tiene traccia del tempo durante il quale un task è attivo, sia in esecuzione sulla CPU, sia in attesa nella runqueue (la coda dei processi pronti per essere eseguiti), calcolando una stima dinamica e progressiva del suo utilizzo della CPU.
PELT infatti si basa su una media mobile pesata esponenzialmente (in inglese chiamata EWMA o Exponentially Weighted Moving Average): questo tipo di media dà maggiore peso agli eventi più recenti e diminuisce progressivamente (in maniera in realtà esponenziale) l'influenza degli eventi più vecchi.

Questo significa che il contributo di un evento (ad esempio, un task che utilizza la CPU) diminuisce col passare del tempo.
Infatti l'algoritmo aggiorna periodicamente il carico del task ogni 1024 microsecondi (circa un millisecondo) e, su un periodo di 32 millisecondi, contribuisce alla metà del

---

[28] Governors, https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, 2024

[29] Il processo di stima del carico di lavoro, il calcolo della frequenza, il concetto di invarianza di frequenza sono descritti nella documentazione di Linux nella sezione “Schedutil Governor" Schedutil Governor, https://www.kernel.org/doc/html/latest/scheduler/schedutil.html, 2024

valore complessivo della stima di utilizzo della CPU: Questo decadimento rapido permette a PELT di essere altamente reattivo ai cambiamenti nel carico di lavoro.

Mostrando un esempio, immaginiamo di monitorare il carico di un possibile task con PELT. Il task originariamente ha un certo carico e diventa inattivo. Il carico di tale task a 32 ms dopo l'inattività scende a 50% del carico originale, e dopo 64 ms di inattività scende al 25% del carico originale.

Calcolo della frequenza

Una volta stimato l'utilizzo attuale con PELT (di cui il valore che useremo nella prossima formula sarà chiamato come *util*), la frequenza della CPU viene calcolata attraverso la semplice formula $f = 1.25 \times f_0 \times \frac{util}{max}$

Dove:

f0 è la frequenza massima della CPU,

util rappresenta l'utilizzo attuale calcolato da PELT,

max è il massimo teorico dell'utilizzo (solitamente 100%)

Invarianza di frequenza

Per evitare incoerenze nella stima e nel calcolo della frequenza è necessario un metodo per normalizzare il valore di carico affinché sia proporzionale alla capacità massima della CPU e non alla frequenza attuale: infatti il calcolo di carico dipende dalla frequenza stessa del processore che in Schedutil dipende a sua volta dalla frequenza del processore, fenomeno che può portare a una gestione sbagliata delle risorse.

Per esempio, il 50% di utilizzo di una CPU a 1 GHz non è equivalente al 50% di utilizzo a 2 GHz. Inoltre, nei sistemi con architetture asimmetriche come big.LITTLE (con core a bassa e alta potenza), l'uso di un core LITTLE al 50% non è equivalente a usare un core big al 50%.

Quindi quando un task è in esecuzione, PELT utilizza una metrica che tiene conto non solo di quanto tempo il task è in esecuzione, ma anche della frequenza alla quale la CPU

sta funzionando, in pratica il carico calcolato dal PELT è scalato per fare in modo che rifletta il carico a massima frequenza

Tale metrica di scalo per architettura più semplice è definita dalla banale formula

$r_{dvfs} = f_{cur}/f_{max}$

Dove *f_cur* è la frequenza corrente e *f_max* è la frequenza massima.

3.5 Implementazione di Schedutil nel kernel Linux

Prima di vedere come funziona il cambio di frequenza con il governor Schedutil in Linux, dobbiamo fare una rigorosa premessa.

Il processo e le funzioni che vedremo in questo capitolo e descritte sempre nella documentazione di Linux riguardante i governor[30] si applicano soltanto per il settaggio della frequenza su processori con valori di frequenza fissi.

I processori con valori di frequenza fissi sono processori che permettono una regolazione in modo coarse-grained, ovvero le frequenze che ammette la CPU sono soltanto alcuni valori di frequenza limitati e predefiniti (per esempio, 1 GHz, 1.5 GHz, 2 GHz, ecc.)

Questa caratteristica era comune nei processori più datati o ancora adesso nei dispositivi a basso costo/consumo.

Per i processori più moderni, verrà utilizzato un altro processo che vedremo successivamente.

Gli scheduler CFS (Completely Fair Scheduler) e Real-Time (RT) implementano entrambi nel loro codice e chiamano la funzione *cpufreq_update_util()*[31] durante alcuni momenti del cambiamento di carico di lavoro.

La funzione *cpufreq_update_util()* attraverso il puntatore a funzione func della struct update_util_data (inizializzata da *sugov_start()*[32]) chiama una funzione di callback

[30] Governors, https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, 2024

[31] Linux kernel 6.12, 2024, File 'sched.h', https://github.com/torvalds/Linux

[32] Linux kernel 6.12, 2024, File 'cpufreq_schedutil.c', https://github.com/torvalds/Linux

che dipende dal tipo di CPU: per processori single core abbiamo *sugov_update_single_freq()*, nel caso in cui un gruppo di CPU condivide la stessa frequenza invece abbiamo *sugov_update_shared();* entrambe fanno cominciare il processo per il cambio di frequenza.

Prima di applicare la frequenza, è necessario sapere quale e calcolare quale applicare. Le logiche dietro questo calcolo le abbiamo viste nei capitoli precedenti, quindi vediamo semplicemente quale è la funzione che si prende responsabilità del calcolo: ovvero *get_next_freq()*[33]: il suo compito come scritto dalla documentazione è quello di calcolare la nuova frequenza della CPU basandosi sull'utilizzo corrente della CPU e su altri parametri come la capacità massima della CPU.

Infine per l'applicazione della frequenza, sono possibili due chiamate che portano a due conclusioni di processo differenti:
La prima è attraverso un processo di target. Questo processo permette al governor di specificare direttamente una frequenza target in hertz. Il driver CPUFreq cercherà di impostare la CPU alla frequenza più vicina supportata dall'hardware.

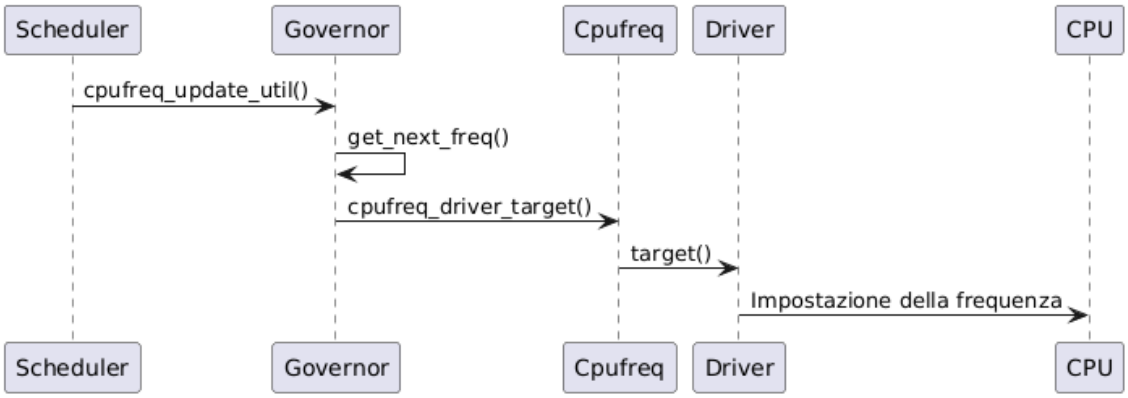

Fig 3.1 Applicazione della frequenza attraverso target

[33] Linux kernel 6.12, 2024, File 'cpufreq_schedutil.c', https://github.com/torvalds/Linux

In Linux l'implementazione di questa funzione è *cpufreq_driver_target()*[34] che consente di specificare direttamente una frequenza target in hertz, tale funzione userà il callback del driver puntato da `target` per applicare la frequenza. (Fig 3.1)
La seconda è attraverso il processo di fast_switch. Questa è una variante ottimizzata che consente cambi di frequenza molto rapidi con il minimo overhead, ideale per rispondere rapidamente a variazioni del carico di lavoro.
Il fast switch infatti è una funzionalità (che troviamo presente nel framework CPUFreq del kernel Linux) che permette di cambiare la frequenza della CPU in modo molto rapido e con un minimo overhead, utile per un governor come Schedutil che richiede frequenti e quindi possibilmente veloci cambi di frequenza dati come già più volte ribadito da variazioni di carico di lavoro.
Tale processo viene implementato in Linux con la funzione *cpufreq_driver_fast_switch()*[35] che ci conferma dalla sua descrizione che permette cambi di frequenza rapidi e aggiunge che è una funzionalità che deve essere supportata dal driver. Anche tale funzione userà il callback del driver puntato da `fast_switch` per applicare la frequenza. (Fig 3.2)

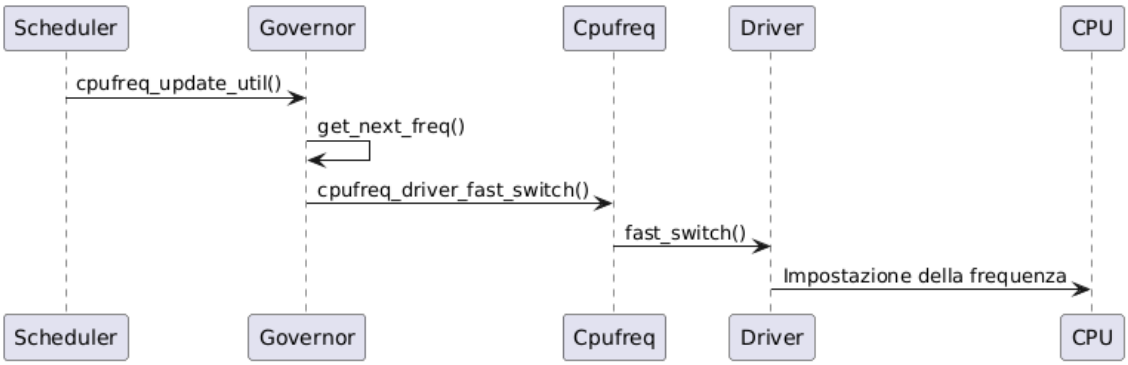

Fig 3.2 Applicazione della frequenza attraverso fast_switch.

[34] Linux kernel 6.12, 2024, File 'cpufreq.c', https://github.com/torvalds/Linux

[35] Ibdem.

3.6 Cambio di frequenza nei processori moderni

Abbiamo visto il cambiamento di frequenza nei processori con valori di frequenza fissi. Troviamo anche un altro tipo di processore ovvero con solo limiti limiti di frequenza, che troviamo ormai come standard nei prodotti moderni.
Nei processori con solo limiti limiti di frequenza la frequenza può essere regolata per valori interni a un limite basso ed un limite alto in modo fine-grained, ovvero con incrementi piccoli e precisi, adattandosi in modo molto accurato alle esigenze del carico di lavoro corrente, piuttosto che fare grandi salti tra poche frequenze predefinite.

La regolazione di frequenza in questo caso non passa né attraverso un governor né attraverso il kernel, ma viene scelta e ragionata in base a delle politiche interne implementate direttamente nel driver.
L'utente o il sistema operativo può agire solamente scegliendo e applicando una delle politiche del processore. L'applicazione avviene tramite la funzione *cpufreq_set_policy()*[36] che chiama a sua volta il callback del driver puntato da `setpolicy`.[37] (Fig 3.3)

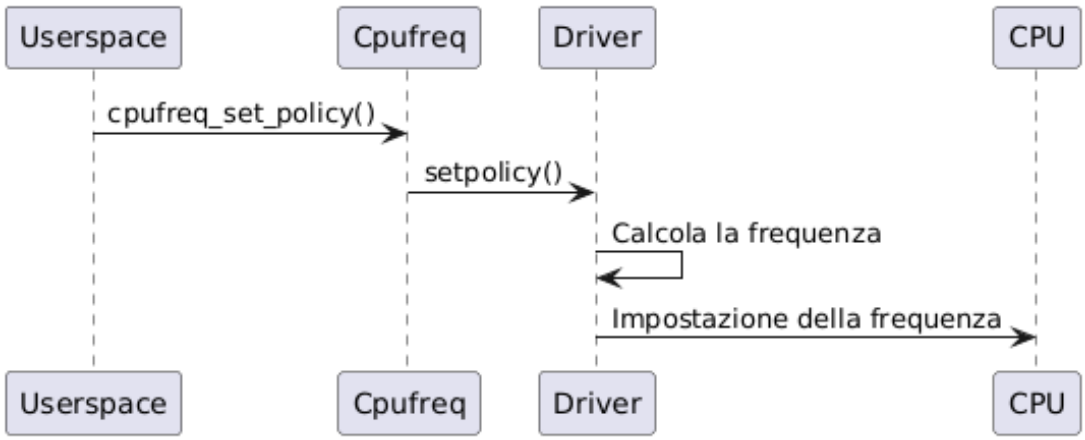

Fig 3.3 Applicazione della frequenza nei processori moderni

[36] Ibdem.

[37] Governors, https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, 2024

4 Device Power Management in Linux: runtime power management

In questo capitolo andremo a vedere il runtime power management, un meccanismo che permette il risparmio di energia elettrica attraverso l'autosospensione dei singoli dispositivi al momento inutilizzati. A fine capitolo vedremo inoltre una spiegazione del funzionamento del meccanismo e una possibile implementazione.

4.1 Runtime power management

Il runtime power management (RPM) è una funzionalità avanzata del kernel Linux progettata da Rafael J. Wysocki per ridurre il consumo energetico dei dispositivi mentre il sistema è attivo.[38]

Questa tecnica permette infatti ai dispositivi di entrare in stati di basso consumo durante l'esecuzione normale, ripristinando la piena operatività quando richiesto.

Come il nome suggerisce, il runtime power management infatti consente ai dispositivi individuali di entrare in stati di basso consumo dinamicamente durante il normale funzionamento del sistema, quando inattivi, senza interferire con altre attività del sistema globale e mentre il sistema continua a funzionare regolarmente.

RPM è composto da un insieme di funzioni[39] che permettono di gestire tali funzionamenti, sfruttando una serie di callback (di sospensione e ripresa) definiti nei driver dei dispositivi e attraverso alcune funzioni e strutture presenti nel modulo pmcore.

---

[38] In questa sezione le informazioni sono state ricavate sia dalla documentazione di Linux "Runtime Power Management" sia da "Adding Runtime Power Management Capabilities to Device Drivers"

Runtime Power Management, https://www.kernel.org/doc/html/latest/power/runtime_pm.html, 2024

Patel, S., 2023, Adding Runtime Power Management Capabilities to Device Drivers, *Embedded Open Source Summit 2023*, June 26-30, Prague, Czech Republic.

[39] codice delle funzioni presente in drivers/base/power/runtime.c del sorgente Linux

Il runtime power management ci permette sostanzialmente quindi di mettere a riposo i dispositivi al momento non utilizzati, facendo risparmiare l'energia che eventualmente sarebbe stata sprecata se inutilmente attivi, però allo stesso tempo cercando di ripristinarli velocemente nel caso di nuovo utilizzo.

4.2 Gestione dei dispositivi inattivi in Linux: autosospensione

Inoltriamoci all'interno dell'implementazione[40] di RPM in un dispositivo e vediamo il funzionamento dell'autosospensione.

Innanzitutto è necessario implementare e attivare il meccanismo di autosospensione di RPM in un dispositivo all'interno del suo driver:
Per fare questo nella funzione di probe del driver (ovvero una funzione di rilevamento e di inizializzazione) devono essere aggiunte tre funzioni che essenzialmente collegano il dispositivo a RPM e attivano la funzione di autosospensione.
Più nello specifico dobbiamo aggiungere:

*pm_runtime_set_active(struct device *dev)*[41]

Che attiva la funzionalità di rpm del dispositivo

*pm_runtime_use_autosuspend(struct device *dev)*[42]

Che abilita la funzionalità di autosospensione del dispositivo

---

[40] Tutto il processo di implementazione è preso dalla conferenza "Adding Runtime Power Management Capabilities to Device Drivers"
Patel, S., 2023, Adding Runtime Power Management Capabilities to Device Drivers, *Embedded Open Source Summit 2023*, June 26-30, Prague, Czech Republic.

[41] Linux kernel 6.12, 2024, File 'drivers/base/power/runtime.c', https://github.com/torvalds/Linux

[42] Ibdem.

*pm_runtime_set_autosuspend_delay(struct device *dev, int delay)*[43]

Che configura il dispositivo ad andare in sospensione dopo un certo delay espresso in millisecondi.

Nelle funzioni di "azione" del driver (che può essere per esempio una funzione di lettura o scrittura), bisogna invece aggiungere altre due funzioni:
Se il dispositivo è in procinto di eseguire una operazione allora bisogna aggiungere ed eseguire la funzione

*pm_runtime_resume_and_get(struct device *dev)*[44].

Tale funzione permette il se necessario riavvio del dispositivo da un possibile stato di sospensione

Se il dispositivo invece ha finito di eseguire delle operazioni ed è diventato quindi inattivo allora:

*pm_runtime_mark_last_busy(struct device *dev)*[45]

che è una funzione che registra il momento in cui è stata per l'ultima volta eseguita una funzione del dispositivo

pm_runtime_put_autosuspend(dev)[46] che è una funzione che prima controlla se il dispositivo non è utilizzato da altri utenti, e in caso positivo (quindi non è più utilizzato da nessun utente) fa partire un timer per l'autosospensione che una volta terminato esegue la funzione *__pm_runtime_suspend()* che porta alla funzione *rpm_suspend()* del runtime Power Management.

---

[43] Ibdem.

[44] Linux kernel 6.12, 2024, File 'pm_runtime.h`, https://github.com/torvalds/Linux

[45] Ibdem.

[46] Ibdem.

Per quanto riguarda il funzionamento, come abbiamo visto precedentemente quando il dispositivo diventa inattivo al termine del timer di autosospensione viene chiamata la funzione *rpm_suspend()*[47].
Tale funzione controlla innanzitutto se il dispositivo può essere sospeso, ed esce e restituisce errore in caso contrario.
Dopodiché il processo di sospensione può essere a discrezione del sistema operativo continuato in maniera asincrona (mettendola in una coda di lavoro) o eseguita direttamente dallo stesso processo eseguendo il callback puntato da *runtime_suspend()*.
In questo processo di callback viene chiamata la funzione di sospensione del driver puntato dall'istanza del dispositivo di tipo dev_ops_pm, struttura definita dal pmcore.
Chiamata la funzione di sospensione del driver verrà avviata la sospensione del dispositivo a livello hardware, che rimarrà sospeso in attesa di essere risvegliato.

[47] Linux kernel 6.12, 2024, File ‘runtime.c`, https://github.com/torvalds/Linux

5 Conclusioni

Nonostante tutti i meccanismi di gestione del consumo energetico in Linux precedentemente analizzati, al momento il sistema operativo non tiene il confronto sul fronte del consumo energetico rispetto alla concorrenza come Windows e MacOs.

Tralasciando MacOS che è un sistema operativo che si dedica a limitate e proprie specifiche architetture, anche il confronto con un sistema operativo versatile come Windows non è proprio favorevole:

Secondo il paper "Comparative Analysis of Power Consumption of the Linux and its Distribution Operating Systems vs Windows and Mac Operating Systems."[48] dopo il confronto rispetto a Windows e MacOS gli autori giungono alla conclusione che le che le distribuzioni Linux consumano più energia rispetto alla concorrenza. Secondo loro Windows, grazie a un forte supporto da parte dei produttori hardware, e MacOS, con la sua stretta integrazione hardware-software, offrono un'efficienza energetica superiore rispetto a Linux, principalmente a causa della mancanza di driver ottimizzati e di moduli di gestione energetica nel kernel di Linux.

Andando ad approfondire il tema ottimizzazione driver, notiamo una particolare e grossa discrepanza di confronto tra consumi soprattutto per quanto riguarda le schede video.

Dal sito "Windows vs. Ubuntu: A Comparative Analysis of Power Consumption"[49] che confronta Windows e il sistema operativo Linux Ubuntu vediamo che su scheda video Nvidia con driver open source Nouveau[50] come i risultati sono particolarmente discrepanti (a sfavore di Linux) sul consumo energetico sia in idle che

---

[48] Najmuddin S., Atal Z., Ziar R.A., 2021, Comparative Analysis of Power Consumption of the Linux and its Distribution Operating Systems vs Windows and Mac Operating Systems. *KJET*, Volume 3, Issue 1, pp. 96–109.

[49] Windows vs. Ubuntu: A Comparative Analysis of Power Consumption, https://algiegray.prose.sh/Windows%20VS%20Ubuntu%20:%20Power%20consumption%20comparison, 06/04/24

[50] driver open source delle schede video Nvidia mantenuto principalmente dalla comunità https://nouveau.freedesktop.org/

sotto stress. (Fig 5.1)

Migliore è la situazione con i driver closed Nvidia dove i confronti con attività semplici o idle hanno dato risultati simili, invece per operazioni più pesanti e orientate alla scheda video ci sono ancora non indifferenti discrepanze. (Fig 5.2)

Con le schede grafiche AMD, che hanno driver open source integrati nel kernel Linux e un pieno supporto da parte della azienda, invece lo stesso sito ci conferma che non ci sono rilevanti discrepanze nel confronto sul consumo energetico, suggerendo che il problema sia più pronunciato con le schede Nvidia.

La conclusione più naturale è che per migliorare il consumo energetico su Linux in sistemi informatici sempre più orientati alla scheda video si debba puntare su un miglioramento dei driver della GPU.

Un approccio suggerito potrebbe essere una progressiva apertura totale delle aziende verso il mondo e la comunità open-source, rilasciando i driver come open source, fornendo risorse, codice per lo sviluppo e la manutenzione di driver insieme a una dettagliata documentazione sulle specifiche hardware come hanno fatto aziende come AMD e Intel, e come si sta lentamente progressivamente approcciando Nvidia[51], oltre ad aumentare i finanziamenti e gli investimenti in progetti Open Source.

[51] Proven L., 2024, Nvidia's next Linux driver to be... just as open, https://www.theregister.com/2024/07/18/nvidia_drivers_remain_as_foss/?td=amp-keepreading

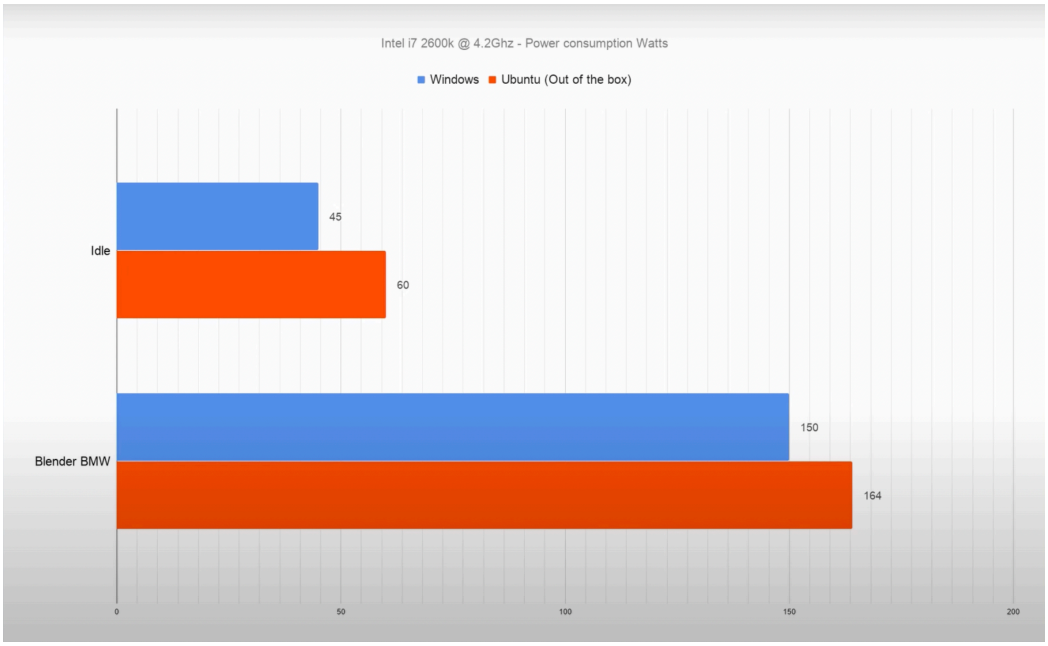

Fig 5.1 Comparazione di consumo energetico tra Windows e Linux con driver open source Nouveau

Fonte: Windows vs. Ubuntu: A Comparative Analysis of Power Consumption, 2024

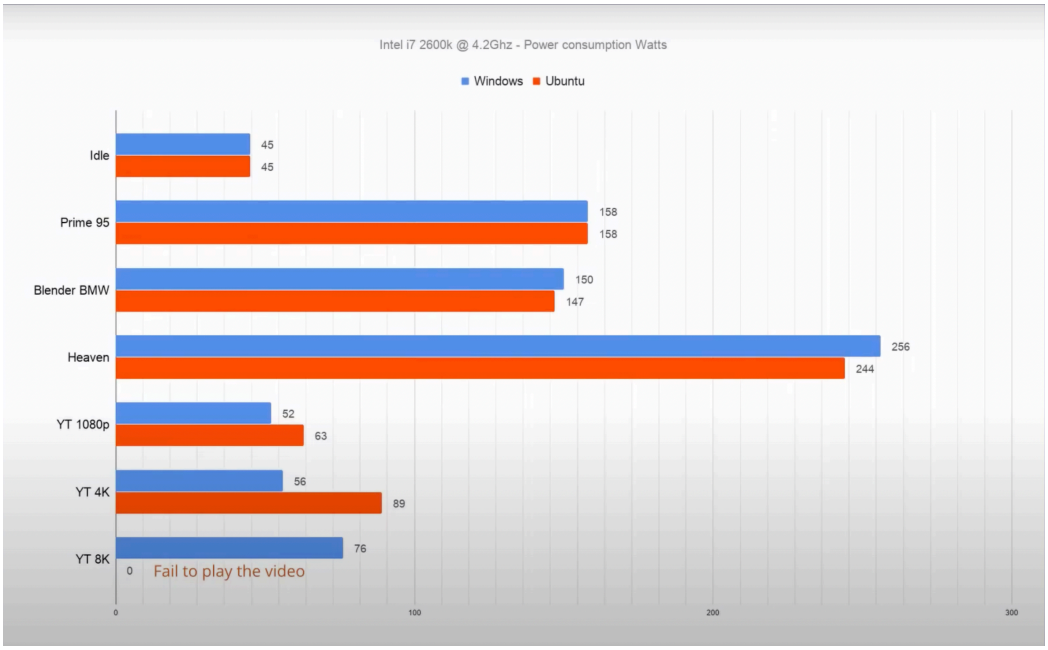

Fig 5.2 Comparazione di consumo energetico tra Windows e Linux con driver Nvidia proprietari

Fonte: Windows vs. Ubuntu: A Comparative Analysis of Power Consumption, 2024

Bibliografia

ACPI Specification 6.4, https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/ 04_ACPI_Hardware_Specification/ ACPI_Hardware_Specification.html, 2024

ACPI Specification 6.5, https://uefi.org/specs/ACPI/6.5/16_Waking_and_Sleeping.html, 2024

Bassignana F., 2017, Gestione dell'alimentazione, parte 1: definizione degli stati ACPI - WEEE Open, https:// weeeopen.polito.it/gestione-dellalimentazione-parte-1-definizione-degli-stati-acpi/

Brown L., Keshavamurthy A.,Li D.S., Moore R., Pallipadi V., Yu L., 2005, ACPI in Linux, Proceedings of the Linux Symposium, vol. 1, pp. 51-69

Cos'è la gestione dell'energia? | IBM, https://www.ibm.com/it-it/topics/energy-management, 2024

CPUfreq - Red Hat Enterprise Linux 7 Power Management Guide, https://docs.redhat.com/en/documentation/ red_hat_enterprise_Linux/7/html/power_management_guide/cpufreq_governors#cpufreq_drivers, 2024

Desktop Operating System Market Share Worldwide | Statcounter Global Stats, https:// gs.statcounter.com/os-market-share/desktop/worldwide/, 14/10/24

Dynamic Power dissipation in CMOS - VLSI UNIVERSE, https://www.vlsiuniverse.com/dynamic- power-dissipation-in-cmos, 21/7/21

Governors, https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, 2024

Linux kernel 6.12, 2024, https://github.com/torvalds/Linux

Linux Statistics Statistics: Market Data Report 2024, https://worldmetrics.org/Linux-statistics/, 23/7/24

Najmuddin S., Atal Z., Ziar R.A., 2021, Comparative Analysis of Power Consumption of the Linux and its Distribution Operating Systems vs Windows and Mac Operating Systems. KJET, Volume 3, Issue 1, pp. 96–109

Nouveau: Accelerated Open Source driver for nvidia cards, https://nouveau.freedesktop.org/

Patch submission for the Linux ACPI project, https://patchwork.kernel.org/project/Linux-acpi/ patch/ 4627718.FT18d2LR5p@vostro.rjw.lan/, 2024

Patel, S., 2023, Adding Runtime Power Management Capabilities to Device Drivers, Embedded Open Source Summit 2023, June 26-30, Prague, Czech Republic

Proven L., 2024, https://www.theregister.com/2024/07/18/nvidia_drivers_remain_as_foss/?td=amp-keepreading

Power management/Suspend and hibernate - ArchWiki, https://wiki.archLinux.org/title/ Power_management/ Suspend_and_hibernate, 2024

Runtime Power Management, https://www.kernel.org/doc/html/latest/power/runtime_pm.html, 2024

Schedutil Governor, https://www.kernel.org/doc/html/latest/scheduler/schedutil.html, 2024

Tringali L., Storia di GNU Linux, *GNU Linux Magazine Italia*, https://www.gnuLinuxmagazine.it/2020/07/ storia- di-gnu-Linux/, 28/7/20.

Windows vs. Ubuntu: A Comparative Analysis of Power Consumption, https://algiegray.prose.sh/ Windows%20VS%20Ubuntu%20:%20Power%20consumption%20comparison, 06/04/24

Ringraziamenti

Desidero esprimere la mia profonda gratitudine a tutte le persone che mi hanno supportato lungo il cammino:

In primo luogo, ringrazio il mio relatore, Prof. Mauro Migliardi, per la sua guida preziosa, per la disponibilità e per i consigli che hanno reso possibile questo lavoro.

Un grazie speciale va ai miei genitori, che mi hanno sostenuto in ogni momento, con amore e fiducia incondizionati. Senza il loro appoggio, nulla di questo sarebbe stato realizzabile.

Infine, vorrei ringraziare i miei amici, con cui ho condiviso fatiche, risate e conquiste. La loro presenza e il loro aiuto ha reso non solo possibile ma anche più leggero e piacevole ogni passo di questo percorso.

A tutti voi, il mio più sincero grazie.