



Descrizione tecnica

NVIDIA Unified Driver Architecture

Un approccio elegante
alla riduzione del TCO



NVIDIA Unified Driver Architecture: unione efficiente di hardware e software

La NVIDIA® Unified Driver Architecture (UDA) funge da base per i driver grafici pluripremiati, le GPU e i processori di piattaforma nForce di NVIDIA.

Una straordinaria qualità di progettazione e un incrollabile impegno aziendale nei confronti della stabilità del software e delle prestazioni hanno reso UDA la sola architettura driver in grado di offrire la massima compatibilità con le soluzioni sia precedenti che future — un solo driver è in grado di supportare tutte le versioni correnti e precedenti dell'hardware. Quando viene resa disponibile una nuova versione di un processore NVIDIA, il driver NVIDIA esistente semplifica l'integrazione del nuovo hardware o sistema in ambienti con numerosi PC desktop e sistemi operativi eterogenei. La NVIDIA UDA non compromette le prestazioni dell'hardware di piattaforma o grafico, sia nuovo che preesistente, ma al contrario riduce il carico di operazioni software e incrementa la funzionalità del driver.

Il successo della NVIDIA UDA continua a essere citato come uno dei fattori principali che induce i maggiori OEM e vendor di sistemi di tutto il mondo a preferire GPU e processori di piattaforma NVIDIA. Grazie alla UDA, tutti gli utenti finali possono sfruttare a fondo la straordinaria flessibilità di aggiornamento ogni volta che siano disponibili nuovi driver. La tradizionale e impeccabile compatibilità con qualsiasi soluzione, sia precedente che successiva, permette inoltre importanti riduzioni del costo totale di proprietà (TCO), risparmi concreti che possono essere trasferiti ai clienti aziendali di grandi dimensioni nonché alle piccole e medie imprese. Numerose caratteristiche della UDA contribuiscono alle riduzioni del TCO:

- ❑ **Riduzione del tempo e del costo di manutenzione:** con una singola versione dei driver NVIDIA è possibile gestire, configurare e installare una sola immagine software anche per le implementazioni di sistemi su vasta scala.
- ❑ **Protezione degli investimenti:** NVIDIA continua ad aggiungere funzionalità e ad ottimizzare le prestazioni del driver UDA. Questo permette agli utenti con sistemi più datati di sfruttare comunque le funzioni più nuove e di incrementare la velocità di esecuzione senza alcun costo aggiuntivo e senza aumentare la complessità per gli utenti finali o i team di manutenzione.
- ❑ **Eliminazione dei conflitti tra hardware e driver:** ogni driver è testato in modo esaustivo con tutti i prodotti precedenti: i nuovi prodotti vengono testati anche con i vecchi driver. Il testing produce operazioni più stabili durante l'intera vita operativa della soluzione.
- ❑ **Maggiore scalabilità:** le operazioni di aggiornamento dei prodotti e di aggiunta di nuovo hardware sono nettamente semplificate. I manager IT possono aggiungere alle proprie configurazioni una gamma completa di prodotti NVIDIA, tutti basati sugli stessi driver.

- **Supporto multi-piattaforma di qualità superiore:** dato che il 90% del codice dei driver NVIDIA può essere condiviso tra i sistemi operativi, NVIDIA è in grado di fornire un supporto ai driver stabile e ad elevate prestazioni, senza limitare la gamma di funzioni offerte dalle CPU e dai sistemi operativi.

Questo documento offre una panoramica completa sulle architetture di driver tradizionali, illustra le sfide tecniche e gestionali associate con le architetture tradizionali e presenta l'approccio NVIDIA UDA, che consente di superare agevolmente tutti questi problemi.

La sfida

Per ottenere un'integrazione ottimale tra hardware e sistema operativo (SO), tradizionalmente si è fatto ricorso a driver scritti in modo stratificato, relegando il codice che controlla direttamente l'hardware al livello di astrazione hardware (HAL). Le dipendenze del SO compongono il "back-end" del driver, definito come la sezione "comune" dato che i driver di tutti i dispositivi hardware richiedono una parte o tutto questo codice back-end. (Vedere la figura 1).

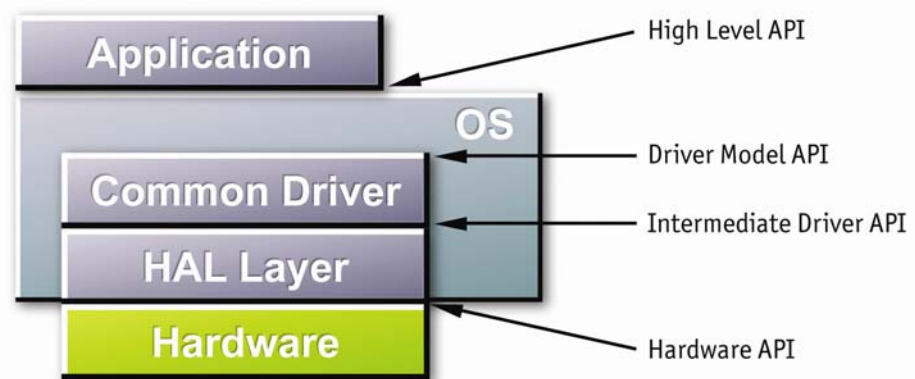


Figura 1. Stack del modello di driver comune

Problemi tecnici

Lo stack del modello di driver comune, pur molto semplice e diretto, è responsabile dell'introduzione di numerosi problemi con l'aumentare della complessità di sistemi e ambienti di computing. I driver di sistema sono influenzati da tre aspetti dell'ambiente di computing:

- ❑ **Le versioni del sistema operativo** quali i vari ambienti operativi Microsoft® Windows® (Windows NT®, Windows 2000, Windows XP, Windows 9x), Linux, e Apple® Mac OS X
- ❑ **Le interfacce di programmazione degli applicativi (API)** ad inclusione di GDI, VPE, WDM, OpenGL® e Microsoft DirectX®
- ❑ **I dispositivi hardware** supportati dalle varie soluzioni di piattaforma e grafiche NVIDIA: passate, presenti e future

Un singolo driver monolitico raggiungerebbe ben presto una dimensione improponibile, con prestazioni decisamente scarse, se dovesse supportare tutte le varie offerte in ciascuna di queste aree. Di conseguenza, l'architettura di driver tradizionale ha offerto numerose versioni dei driver, ciascuna in grado di supportare qualche sottogruppo di SO, API e hardware.

L'applicazione delle consuete tecniche di programmazione e della tradizionale architettura di driver dà luogo a numerosi problemi tecnici:

- ❑ **Dilazione del time-to-market**
Tutti i driver devono essere ottimizzati per supportare SO, API, o soluzioni hardware nuovi o perfezionati. Le numerose combinazioni si traducono in sessioni prolungate di testing e certificazione.
- ❑ **Incremento dei costi di supporto**
Differenti versioni dei driver per i vari SO, API e soluzioni hardware si traducono in un incremento dei tempi di manutenzione IT per gestire combinazioni stabili di hardware e software.
- ❑ **Degrado delle prestazioni**
Diventa virtualmente impossibile ottimizzare ciascun possibile percorso di esecuzione con più driver. L'uso di un solo driver di grandi dimensioni produce un ulteriore rallentamento delle prestazioni.
- ❑ **Scarsa scalabilità**
Un approccio convenzionale alla programmazione non consente una corretta scalabilità data l'elevata complessità di hardware e software. Un driver monolitico incrementa la complessità in modo esponenziale, rendendo impossibile conservare qualità e prestazioni.

Problemi di gestione

Un'analisi di business dell'architettura tradizionale dei driver offre un'altra serie di problemi da risolvere:

- ❑ **Costi di sviluppo del software**
L'uso del modello di driver tradizionale costringe a disporre di un gruppo di tecnici di grandi dimensioni per supportare l'engineering del driver e le soluzioni dei bug per tutti i vari driver. Questi costi vengono poi inevitabilmente trasferiti ai clienti.
- ❑ **Complessità del procedimento di release**
Gli sforzi di QA sono complessi e piuttosto onerosi dal punto di vista del tempo impiegato. Come ulteriore complicazione, le attività di testing devono essere duplicate per ciascuna fase della supply chain: per il vendor di driver, gli OEM e i vendor di sistemi, nonché per le grandi aziende che certificano tutte le attrezzature prima dell'effettiva implementazione.
- ❑ **Fasi di testing della certificazione costose e lente**
La certificazione deve essere portata a termine per ciascuna combinazione driver-hardware e per ciascun sistema che integra il driver. I driver monolitici di grandi dimensioni solitamente prevedono numerosi passaggi di testing e cicli di certificazione di maggiore durata.

La soluzione NVIDIA

Gli architetti NVIDIA hanno deciso di progettare e implementare un'architettura di driver che minimizzasse il TCO per i processori NVIDIA, massimizzasse le prestazioni e offrisse una stabilità e una compatibilità senza paragoni per l'intera linea di processori grafici e di piattaforma. La soluzione adottata è la NVIDIA Unified Driver Architecture: questa soluzione soddisfa tutti gli obiettivi ed è ben presto diventata lo standard più innovativo del settore (vedere la figura 2).

Grazie all'approccio NVIDIA UDA, i driver NVIDIA si concentrano sull'implementazione delle funzionalità delle API invece di tracciare tutte le differenze dell'hardware. Tutte le operazioni di controllo dell'hardware sono gestite tramite il modello di programmazione orientato a oggetti e basato su classi. Le funzionalità sensibili alle prestazioni (funzioni di elaborazione grafica e audio) sfruttano un HAL ad implementazione hardware residente su chip NVIDIA. Le funzionalità ad implementazione hardware minimizzano il carico di lavoro associato con i driver tradizionali. La combinazione hardware/software bilancia in modo eccellente prestazioni e qualità, un risultato che non può essere conseguito con una soluzione che si limita alla sola dimensione software.

Tutti i driver NVIDIA ottengono prestazioni straordinarie pur garantendo la massima compatibilità con tutte le soluzioni della società, sia precedenti che successive. Un driver è in grado di supportare tutte le versioni dell'hardware pertinente con tecniche di programmazione basate su classi ad astrazione elevata che proteggono il software dall'hardware di basso livello.

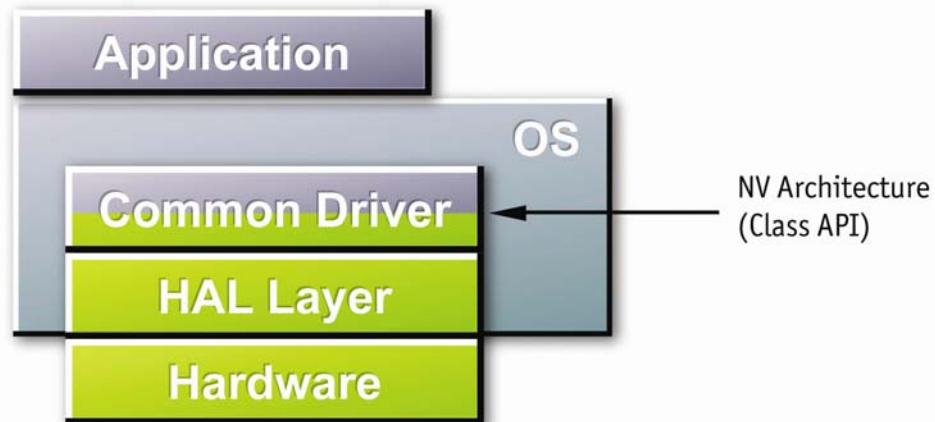


Figura 2. Stack del modello di driver NVIDIA per GPU e funzionalità sensibili alle prestazioni

Programmazione orientata agli oggetti

La NVIDIA UDA include un'astrazione orientata agli oggetti delle funzionalità hardware, anch'essa definita come un modello basato su classi. *Oggetti* e *classi* forniscono meccanismi per il controllo efficiente dell'accesso alle funzionalità e alle informazioni sul contesto. Grazie all'astrazione delle funzionalità del motore, l'architettura NVIDIA mantiene l'indipendenza dal motore pur riducendo anche il carico di lavoro per CPU, memoria e bus dovuto alle funzioni grafiche e di sistema.

Nota: una **classe** è una definizione astratta di una particolare funzione hardware (da implementare come mappatura di registro). Un **oggetto** è una singola istanza di una classe (una specifica mappatura del registro), e quindi contiene un contesto particolare. Un **metodo** è un registro di sola scrittura all'interno di un oggetto.

La struttura di classi NVIDIA UDA e i *metodi* replicano gli effettivi modelli delle API e il loro impiego. Per esempio, il driver delle GPU NVIDIA fornisce classi che sono progettate per corrispondere con esattezza alle recenti interfacce di programmazione di Microsoft Windows.

Gestore delle risorse (Resource Manager)

All'interno con il livello hardware della NVIDIA UDA, il gestore delle risorse NVIDIA fornisce una soluzione intelligente per il riconoscimento e la gestione delle classi e degli oggetti (vedere la figura 3). Il software dei driver NVIDIA fornisce una libreria di livello kernel per il riconoscimento e per la gestione delle classi e degli oggetti. Questa innovazione NVIDIA esegue la gestione dello stato e del contesto e l'assegnazione di tutte le risorse architettoniche secondo le effettive necessità di ciascun client del driver. Si noti che in questo schema di gestione del contesto i client multipli non sono al corrente della presenza di altri utenti concomitanti del chip.

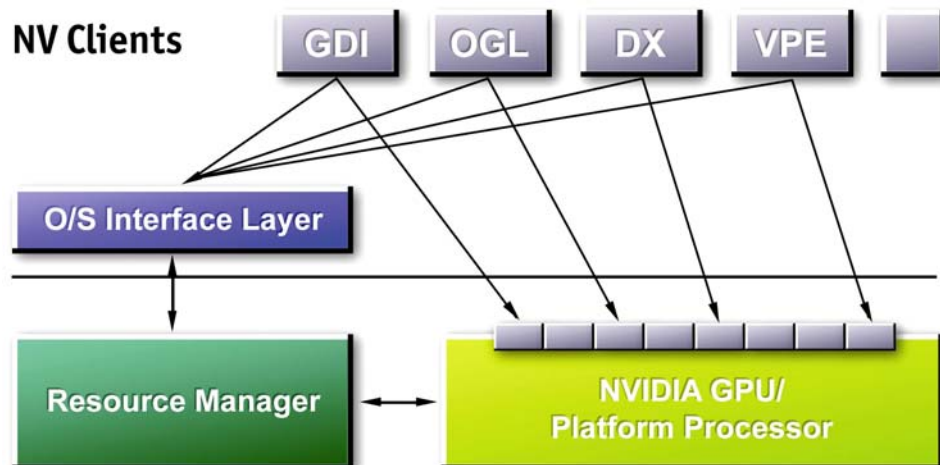


Figura 3. Modello di driver architettivo con il Resource Manager

L'assegnazione e la gestione di *oggetti* architettonici (risorse hardware) fornisce diverse capacità per la gestione della compatibilità. Come descritto in precedenza, il Resource Manager definisce un canale tra l'hardware e un client del driver al fine di minimizzare il carico di operazioni del driver per ottimizzare l'impiego delle risorse hardware.

Grazie al riconoscimento delle classi designate, il Resource Manager è in grado di gestire i casi di eccezione hardware. Le differenze nelle implementazioni hardware sono rese trasparenti per l'utente e gestite in modo da supportare la compatibilità dei driver su tutte le piattaforme. Il Resource Manager permette anche di utilizzare le interfacce di classe legacy per mantenere la compatibilità con le soluzioni precedenti nelle release hardware successive. I tecnici delle applicazioni possono interfacciarsi con tutto l'hardware avvalendosi di queste convenzioni standard, il che significa che il codice legacy non deve essere modificato per supportare il nuovo hardware.

Le altre funzioni gestite dal Resource Manager includono:

- ❑ **Gestione degli errori**
Le capacità di riconoscimento delle classi del Resource Manager offrono un eccellente meccanismo per l'implementazione delle soluzioni dei bug.
- ❑ **Porzioni dipendenti dal SO dell'architettura**
Il Resource Manager può inoltre essere utilizzato per gestire le differenze tra release e release per la gestione della memoria SO, controllo di processo/thread, plug-n-play, gestione dell'alimentazione e altre capacità.

Vantaggi dell'UDA

La NVIDIA UDA è stata messa alla prova con successo su diverse generazioni di tecnologie NVIDIA e numerose release di prodotti NVIDIA sin dal 1998, con l'introduzione della NVIDIA RIVA128™. Gli OEM NVIDIA e i clienti hanno sostenuto entusiasticamente questa soluzione, indicando altri vantaggi:

- ❑ **Progettazione migliore e maggior numero di funzioni incorporate**
Dato che è necessario un numero minore di tecnici per il controllo della versione e la gestione delle release, è possibile assegnare un maggior numero di risorse all'ottimizzazione delle prestazioni e all'aggiunta di nuove funzionalità. L'approccio NVIDIA UDA produce anche una maggiore libertà per quanto riguarda la modifica dell'hardware di livello inferiore, senza alcun impatto sugli OEM e gli utenti finali.
- ❑ **Prestazioni eccellenti**
Le funzionalità dei driver nell'hardware offrono le migliori prestazioni possibili e ottimizzano tutte le applicazioni.
- ❑ **Stabilità e affidabilità**
La NVIDIA UDA ha superato ogni test e contribuisce al record senza paragoni della società per quanto riguarda stabilità e affidabilità di hardware e software.
- ❑ **Netta riduzione del time-to-market**
Le release NVIDIA sono realizzate a tempo di record — meno di 100 giorni dal chip tape-out alla produzione, mentre il resto del settore offre tempi superiori ai sei mesi. Gli OEM ottengono un accesso rapido alla nuova tecnologia, e l'UDA significa anche una semplice installazione e supporto di nuove release, con un solo driver a supportare le soluzioni, sia precedenti che nuove.

❑ **Riduzione del TCO**

I manager IT possono generare una singola immagine software con una sola versione del driver NVIDIA e implementare migliaia di sistemi, ciascuno con processori NVIDIA differenti. Questo consente ai manager IT di qualificare, gestire e supportare una singola architettura software per più configurazioni hardware.

Conclusioni

NVIDIA UDA ha stabilito una nuova tradizione per le architetture driver offrendo le migliori prestazioni, compatibilità senza paragoni e TCO significativamente ridotto. UDA non si limita a offrire un'architettura davvero elegante. I vantaggi di UDA ha un impatto positivo su tutti quelli che scelgono la tecnologia NVIDIA:

- ❑ **Le aziende** possono sfruttare al massimo i risparmi in termini di costo IT e tempi derivanti dalla semplificazione del supporto e delle procedure di aggiornamento.
- ❑ **I manager IT** possono gestire diversi prodotti hardware con gli stessi driver, procedere a effettuare aggiornamenti a nuove piattaforme senza cambiare i driver, ridurre i tempi di rollout del sistema e supportare configurazioni flessibili delle piattaforme hardware.
- ❑ **Gli utenti finali** ottengono un accesso rapido a nuovi prodotti e la capacità di eseguire i giochi e le applicazioni esistenti sulle nuove piattaforme.
- ❑ **Gli sviluppatori software** possono scrivere una sola versione di un'applicazione ottimizzata per tutte le soluzioni NVIDIA.
- ❑ **I vendor di sistemi** possono offrire ai clienti la possibilità di generare una singola immagine software che supporta un'intera linea di hardware, semplificando l'introduzione di nuove piattaforme in una base installata, snellendo i cicli di certificazione e qualifica e offrendo un TCO complessivo migliore ai propri clienti.

Le soluzioni NVIDIA continueranno ad affidarsi alla UDA per consentire la massima rapidità di distribuzione di driver software ad alta qualità e altissime prestazioni. La UDA supera le sfide associate con le architetture di driver tradizionali del passato e permette la rapida introduzione di innovazioni hardware senza impatti negativi sulla compatibilità con sistemi e applicazioni esistenti. Nessun altro vendor di processori di piattaforma e GPU è in grado garantire la possibilità di impiegare una sola immagine software sull'intera gamma di prodotti della società.

Notifica

TUTTE LE SPECIFICHE DI PROGETTAZIONE NVIDIA, LE SCHEDE DI RIFERIMENTO, I FILE, I DISEGNI, LA DIAGNOSTICA, LE LISTE E ALTRI DOCUMENTI (UNITAMENTE E SEPARATAMENTE, DEFINITI "MATERIALI") SONO FORNITI NELLO STATO IN CUI SI TROVANO. NVIDIA NON OFFRE GARANZIE, ESPRESSE, IMPLICITE, STATUTARIE O DI ALTRO TIPO IN RELAZIONE AI MATERIALI, E RIFIUTA ESPRESSAMENTE OGNI GARANZIA IMPLICITA DI NON VIOLAZIONE, COMMERCIALIZZABILITÀ E IDONEITÀ A SCOPI SPECIFICI.

Le informazioni fornite sono ritenute accurate e affidabili. Tuttavia, NVIDIA Corporation non si assume alcuna responsabilità per le eventuali conseguenze derivanti dall'uso di tali informazioni o da qualsiasi violazione di brevetti o altri diritti di terze parti che possono conseguire dal loro uso. Non viene concessa alcuna licenza implicita o in altro modo in base a nessun brevetto o diritto di autore di proprietà di NVIDIA Corporation. Le specifiche tecniche menzionate nella presente pubblicazione sono soggette a modifica senza preavviso. Questa pubblicazione rimpiazza e sostituisce tutte le informazioni precedentemente fornite. Non si autorizza l'impiego dei prodotti di NVIDIA Corporation come componenti cruciali di dispositivi per il supporto vitale o per sistemi che non abbiano ricevuto l'espressa approvazione scritta di NVIDIA Corporation.

NVIDIA e il logo NVIDIA sono marchi registrati e RIVA128 è un marchio di NVIDIA Corporation.

Altri nomi di società e di prodotti possono essere marchi o marchi registrati dei rispettivi detentori.

Copyright NVIDIA Corporation 2003



NVIDIA.
NVIDIA Corporation

www.nvidia.it