Quando il software "divorzia" dall'hardware

Una dettagliata analisi dei benefici legati alla virtualizzazione delle componenti hardware e software durante lo sviluppo di un progetto

Jim Turley (*)
technology analyst

Quasi ogni giorno da più parti si levano pressanti richieste finalizzate al cambiamento delle modalità con cui vengono effettuate le operazioni di ingegnerizzazione, programmazione, gestione o controllo qualità. Per contro, la maggior parte degli ingegneri e dei progettisti desidera solamente continuare a svolgere il proprio lavoro. Ovviamente, ciò non significa che i progressi nel campo dei tool e dei linguaggi non siano interessanti, solamente che vengono visti come astratti e non pertinenti. Essi potrebbero essere utili per il prossimo progetto o per quelli successivi, ma al momento attuale rappresentano solo un elemento di distrazione.
Attualmente, l'aspirazione di un programmatore è quella di terminare la scrittura del codice, effettuare il debug, determinare alcune caratteristiche non completamente definite e consegnare il lavoro. Nel momento in cui l'hardware funziona e il codice gira, tutto sta procedendo secondo i canoni stabiliti. Nel momento in cui l'hardware non è pronto, o non funziona correttamente, oppure non rappresenta fedelmente il prodotto finale, le cose tendono inevitabilmente a complicarsi.
Solitamente coloro che si occupano di hardware cercano di realizzare prototipi mentre, dal canto loro, gli sviluppatori software sono impegnati nella scrittura dell'hardware che è in via di sviluppo. Non è possibile scrivere software per un hardware che non esiste e neppure effettuare il debug dell'hardware senza disporre del relativo software diagnostico. Una situazione tipica è quella in cui i programmatori sono in attesa dell'hardware e i progettisti del codice funzionante. La cooperazione, passo dopo passo, delle due parti in causa porta, alla fine, alla realizzazione di un hardware sufficientemente stabile da permettere lo sviluppo del software e viceversa.

Teoria e pratica

Quella sopra delineata non è la metodologia da seguire. L'approccio corretto richiede l'apporto di qualche modifica. Si faccia l'ipotesi, puramente teorica, che i programmatori non abbiano bisogno dell'hardware per scrivere codice specifico. Si presuma anche che l'hardware non sia effettivamente necessario per realizzare un prodotto (alcuni esperti di marketing asseriscono che ciò già accade: l'hardware è generico e il prodotto viene definito dal software. Telefoni cellulari, lettori DVD e iPod sono ottimi esempi di prodotti realizzati su un hardware generico che si differenziano in base al software).
A livello teorico ogni programmatore potrebbe scrivere qualsiasi tipo di codice – dai sistemi operativi ai driver dei dispositivi, ai gestori degli interrupt, ai programmi applicativi, senza disporre di un hardware affidabile o, anzi, di alcun tipo di hardware. Ogni programmatore sarebbe in grado di creare software senza disporre dell'hardware. Anzi, non solo scrivere software, ma anche effettuare il debug, il test, la profilazione, l'aggiornamento e così via. In breve, si sta facendo l'ipotesi di avere risorse hardware virtuali al posto di un oggetto reale.
Non si tratta di un concetto innovativo e neppure particolarmente originale. I sistemi operativi gestiscono una "memoria virtuale", effettuando lo swapping (ovvero il trasferimento) del codice in entrata e in uscita dalla memoria fisica e regolando i puntatori in modo da far apparire una disponibilità di memoria superiore a quella effettiva. Anche i driver dei dispositivi vengono "virtualizzati" in modo che i dischi si comportino allo stesso modo anche se hard drive, floppy, thumb drive e CD-ROM non hanno quasi niente in comune tra di loro.
Il linguaggio Java, ad esempio, è basato su un hardware immaginario che ipotizza l'esistenza di una macchina virtuale Java (JVM – Java Virtual Machine), ovvero un computer che non è mai stato realizzato. Milioni di computer fanno girare codice Java anche se non è mi esistita una JVM reale, ma solamente un'approssimazione software. Nonostante la sua complessità, Java gira su milioni di macchine differenti che non hanno tra di

loro nulla in comune. Persino i sistemi operativi, che sono ottimizzati per l'uso su singole piattaforme hardware, sono stati virtualizzati. Un livello di astrazione hardware (HAL – Hardware Abstraction Layer) isola il kernel del sistema operativo e i driver dall'hardware effettivo.

Fino a questo momento ci si è basati su solide fondamenta teoriche. Qualsiasi sistema hardware può essere "clonato" per mezzo del software con un grado di fedeltà sufficiente da garantire le medesime funzionalità dell'originale. Come nel caso della classica macchina di Turing, un utente non è in grado di distinguere una simulazione dall'oggetto reale. Paradossalmente, il clone software a volte è più veloce dell'hardware che è chiamato a emulare, com'è accaduto ad esempio nel caso di Transmeta con il suo chip Crusoe Pentium compatibile. Il software di traduzione per l'architettura x-86 di Dec potrebbe eseguire alcune istruzioni hardware più velocemente di un microprocessore Intel. La lista potrebbe continuare, ma l'aspetto da tenere in considerazione è che questi "sostituti" software possono vantare una lunga e gloriosa tradizione.

Le società utilizzano la virtualizzazione software per ragioni commerciali, per avere accesso al software esistente, per semplificare lo sviluppo software e per conferire maggiore portabilità ai programmi. Altre invece lo fanno per guadagnare tempo – e queste considerazioni riconducono all'interno dei laboratori.

Alcuni esempi

Se è possibile eseguire l'astrazione del sistema hardware, le cause di molti problemi sono destinate a scomparire con esso. La programmazione non cambia, ma il debug risulta molto diverso. Qualsiasi risorsa hardware (registro, locazione di memoria e così via) è visibile, collaudabile e adattabile. L'acquisizione dati diventa un compito banale e il canale ad ampiezza di banda limitata verso l'emulatore, il debugger o il buffer di trace semplicemente scompare. I breakpoint possono essere comunque complessi, in quanto relativi a un modello realizzato secondo le proprie necessità.

In media lo sviluppatore impiega il 40% del proprio tempo nelle operazioni di debug – un tempo superiore a quello richiesto da qualsiasi altro compito, compresi la programmazione, la scrittura delle specifiche e l'ingegnerizzazione. Nel caso fosse possibile ridurre, anche di poco, il tempo necessario per il debug, l'investimento fatto nella realizzazione del modello sarebbe sicuramente ripagato.

La generazione del modello virtuale rappresenta l'unica novità per molti sviluppatori, mentre tutte le altri fasi del processo non subiscono variazioni. Il modello stesso solitamente serve come specifica di prodotto per i team che si occupano dello sviluppo hardware e software. Si tratta di una specifica eseguibile "dal vivo" invece che un semplice documento cartaceo. Un modello di questo tipo risulta particolarmente utile per le aziende di grandi dimensioni i cui team di sviluppo sono dislocati in differenti aree geografiche.

Con l'astrazione dell'hardware, è inutile mettere a punto sistemi hardware complessi e sofisticati che non hanno alcun significato per coloro che si occupano dello sviluppo del software.

Se l'hardware non esiste, non esisteranno neppure difetti hardware. Sebbene possa sembrare semplicistica, questa affermazione è più concreta di quanto si possa pensare. Dopo tutto, i programmi scritti in Java non subiscono alcuni tipo di "crash" per ragioni legate all'hardware. Essi potrebbero contenere errori come qualsiasi altro programma, ma non sono soggetti a malfunzionamenti imputabili a incompatibilità a livello hardware. Se un nuovo programma Java è stato completamente testato su un PC, il medesimo programma potrà girare su qualsiasi altra piattaforma Java, indipendentemente dai cambiamenti a livello hardware. In definitiva, non è dunque necessario disporre dell'hardware target, o persino conoscere qualcosa riguardo ad esso. Si immaginino a questo punto i vantaggi legati alla possibilità di sviluppare e far girare codice per un sistema che non è stato realizzato, di cui non si conosce nulla o che risulti appena abbozzato. Questo era certamente l'obiettivo che si erano posti i creatori di Java. Purtroppo questo linguaggio presenta alcuni svantaggi. Il primo, e più ovvio, è che risulta necessario scrivere codice Java, un linguaggio che non è certamente in cima alla lista delle preferenze della maggior parte dei programmatori.

In base ai risultati di una recente indagine, solo il 3% dei programmatori lavorano principalmente con Java, rispetto all'81% che utilizza C o C++. Per molti programmatori il cambio di linguaggio è paragonabile a una conversione religiosa, per cui il passaggio a Java (o a qualsiasi altro linguaggio) appare un evento decisamente improbabile.

La lentezza è un altro problema di Java. Si tratta infatti di un linguaggio interpretato che include un gran numero di operatori complessi e verifiche degli indici, senza dimenticare l'esecuzione della garbage collection durante il run time e alte caratteristiche piuttosto originali. I programmatori in real time evitano accuratamente Java a causa dell'imprevedibilità delle sue prestazione. Inoltre, Java utilizza parecchia memoria. La macchina JVM richiede molta memoria (dell'ordine di alcuni Mbyte), senza dimenticare le dimensioni dello stesso programma Java. Poi c'è lo spazio dello stack. Questo overhead in Java coinvolge il sistema target e non il sistema di sviluppo. Per molti programmatori, Java richiede troppe risorse (in termini di memoria, velocità, complessità e produttività) rispetto a quanto può offrire. Questa è la ragione per cui un numero così limitato di program-

matori ha adottato questo linguaggio nonostante sia sulla scena da oltre un decennio e venga proposto in parecchie versioni.
L'obiettivo sarebbe quello di poter disporre dell'hardware virtuale previsto da Java senza l'overhead in run time sul sistema target. Senza dimenticare che il cambiamento di linguaggio rappresenta un ostacolo quasi insormontabile. In definitiva, lo scopo principale è la virtualizzazione dell'hardware.

I vantaggi dell'hardware virtualizzato

Dal punto di vista teorico, la disponibilità di un hardware virtualizzato potrebbe comportare numerosi vantaggi per i programmatori. Questi sono riassunti di seguito in quello che si può definire una sorta di "manifesto dello sviluppo del software virtualizzato". Per i programmatori sarebbe possibile:
- Sviluppare il software prima della disponibilità dell'hardware target;
- Valutare configurazione hardware alternative;
- Isolare definitivamente gli errori software da quelli hardware;
- Migliorare l'affidabilità nel collaudo eseguito esclusivamente via software e nei relativi risultati;
- Sfruttare l'esperienza acquisita con gli attuali linguaggi, tool e pacchetti software;
- Migliorare la visibilità del debug e la controllabilità rispetto all'hardware effettivo;
- "Liberare" l'hardware effettivo per consentirne il debug;
- Isolare gli effetti delle revisioni effettuate sul silicio o sulla scheda dal processo di debug;
- "Anticipare" la "curva di complessità" per lo sviluppo di nuovi sistemi;
- Ridurre il tempo di debug mediante l'eliminazione delle configurazioni hardware;
- Migliorare la qualità del prodotto grazie alla possibilità di isolare più velocemente la fonte degli errori.

Se si esamina il primo punto, appare chiaro che grazie all'hardware virtualizzato un programmatore può iniziare la scrittura del codice prima che sia disponibile l'hardware target effettivo. Ciò rappresenta intrinsecamente un vantaggio per l'intero team di sviluppo. Nel caso lo sviluppo software possa iniziare con alcune settimane di anticipo senza che sia disponibile l'hardware, si avrebbe una forte incentivazione dal punto di vista finanziario. In questo modo risulta possibile eliminare una causa di contenzioso tra gli ingegneri hardware e software: la condivisione dell'hardware funzionante.
Inoltre, i produttori di chip possono fornire modelli virtuali ai loro potenziali clienti, ai partner software e a tutte le altre entità coinvolte. Nel momento in cui sono disponibili i primi chip, le infrastrutture e il supporto sono già pronti e i primi clienti potranno trarre un sicuro vantaggio dalla disponibilità del software e della logica di supporto.
Quando si parla dei primi prototipi, il software virtualizzato non si preoccupa dell'aspetto dell'hardware, per cui gli sviluppatori potrebbero essere incoraggiati a esplorare differenti configurazioni. Infatti, risulta molto più semplice implementare un modello dimostrativo software di un sistema a quattro processori piuttosto che progettarne e realizzarne uno fisicamente. Oltre a essere decisamente più economico. Se si vuole valutare l'impatto dell'aggiunta di canali Ethernet sul throughput o la variazione delle prestazioni imputabili a una diminuzione della capacità di memoria, è senza dubbio può semplice ricavare queste risposte attraverso la modellazione software piuttosto che mediante la "forzatura" dell'hardware.
Nel momento in cui il software è sviluppato indipendentemente dall'hardware, gli inevitabili errori del codice sono di natura esclusivamente software. Non possono esserci errori hardware per il semplice fatto che l'hardware non esiste. Le piattaforme virtualizzate non sono sensibili alle fluttuazioni dell'alimentazione, a fenomeni di crosstalk o alle tolleranze dei componenti. Esse non variano da lotto a lotto, non hanno collegamenti che si possono "allentare" nei momenti meno opportuni. In breve, l'hardware virtualizzato rappresenta una base stabile e affidabile sulla quale i programmatori possono realizzare il loro codice.

Isolare gli errori

Quando gli errori software sono isolati da quelli hardware, aumenta il grado di affidabilità. Il fatto che il software sia stato esaustivamente e adeguatamente collaudato rappresenta un vantaggio per tutti. Se il codice funziona correttamente sul modello, risulta corretto per definizione. Qualsiasi errore che si manifesta improvvisamente quando il codice gira sull'hardware reale è implicitamente un errore legato all'hardware. Il software è già stato collaudato per cui metà del lavoro di debugging è già stato svolto.
Uno degli aspetti più apprezzati dell'hardware virtuale è rappresentato dal fatto che non ha alcun impatto relativamente alla tool chain software: infatti non sono richiesti nuovi linguaggi, compilatori particolari o macchine virtuali run-time. Ciò rappresenta un indubbio vantaggio per i programmatori, che possono continuare a svolgere il loro lavoro seguendo le modalità tradizionali, utilizzando tool che ben conoscono. I dati di una recente indagine hanno dimostrato che i programmatori sono più "fedeli" ai loro compilatori che non ai loro processori.
Il processo di debug è insieme arte e scienza. Scienza perché bisogna conoscere le modalità per correggere i difetti, arte perché è indispensabile sapere dove andarli a cercare. Le schede di sviluppo sono irte di sonde, fili, commutatori e Led allo scopo di rendere visibile ciò che è celato. Si tratta di una lotta continua tra ciò che il programmatore ha bisogno di vedere e ciò che l'hardware gli consente di vedere. Con l'hardware virtuale, tutti questi problemi sono destinati a scomparire. Un processore, una scheda o un sistema virtuali sono completamente visibili. Non ci sono registri, bus o locazioni di memoria che non possano essere collaudate, modificati e verificati. Tutti gli strata-

gemmi che i programmatori hanno sviluppato nel corso degli anni possono essere abbandonati: si tratta dell'unico caso dove le tecniche software cambiano con l'hardware virtualizzato.

Una volta disponibili le schede di sviluppo hardware, esse possono rimanere presso il team che si occupa dello sviluppo hardware. Ovviamente, essi dovranno eseguire il test e il debug delle loro prime revisioni, mentre i programmatori stanno lavorando nel laboratorio attiguo. Ma in questo caso non devono condividere le loro "preziose" schede di sviluppo. L'hardware può restare presso il team che si occupa del suo sviluppo, in modo da accelerare il ciclo di debug. Ciò si rivela vantaggioso negli stadi finali, durante le fasi di perfezionamento o aggiornamento dell'hardware: l'hardware revisionato continua a rimanere presso il team di sviluppo, mentre i programmatori si cimentano con il modello virtuale aggiornato.

Il debug negli stadi iniziali risulta particolarmente vantaggioso in presenza di chip completamente nuovi e non collaudati come il resto del prodotto. Sia che si tratti di un processore appena introdotto o un nuovo Fpga, i chip sono spesso soggetti a parecchie revisioni, alla stessa stregua di una scheda o di un sistema. I programmatori, dal canto loro, devono occuparsi esclusivamente del debug del codice, senza preoccuparsi del fatto che quello che un giorno funziona perfettamente il giorno seguente non funziona affatto.

Non è certamente un segreto il fatto che i sistemi embedded diventino di giorno in giorno più complessi: in altre parole ciò significa più hardware, più software e tempi di sviluppo più lunghi. Un progetto che preveda hardware e software virtualizzati consente ai team di sviluppo di avere un vantaggio sullo sviluppo del prodotto dell'anno prossimo o sull'evoluzione tecnologica futura. Processori multi-core, nuove periferiche e interfacce della prossima generazione non rappresentano un problema insormontabile per i team di sviluppo che possono disporre di modelli virtuali. Molto spesso un nuovo prodotto viene offerto in differenti configurazioni – hard disk di maggiore capacità, dimensione di memoria inferiore, diverse possibilità di espansione e così via. Ovviamente, i team che si occupano della parte hardware e di quella software devono impegnarsi a collaudare tutte le versioni. Il software, naturalmente dovrà girare su ognuna di esse, adattandosi automaticamente alle caratteristiche che sono o meno abilitate. Di conseguenza risulta necessario effettuare il test su differenti configurazioni hardware, fatto questo che di solito richiede la presenza di un tecnico. Qualunque sia l'operazione richiesta – cambio dell'impostazione dei ponticelli, aggiunta o rimozione di una scheda mezzanino, rimpiazzo di controllori o riconfigurazione dei conduttori – il team di sviluppo software non è attrezzato per eseguirla.

Ancora una volta, se si elimina l'hardware, scompaiono i problemi legati alla sua configurazione. Differenti modelli del sistema target possono quindi essere collaudati in maniera estremamente semplice.

Infine, tutto viene ricondotto all'aspetto qualità. La qualità non può essere collaudata, ma deve essere progettata. Il semplice scarto dei prodotti difettosi non contribuisce a eliminare la causa del problema. Per correggere i difetti è necessario prima individuarli e questa operazione risulta estremamente più semplice e veloce quando è possibile separare la componente hardware da quella software.

Superare le aspettative

Quella appena esposta è una teoria senza dubbio accattivante. Infatti, lo sviluppo della parte software è stata nettamente separato da quello della componente hardware. Allo sviluppo software sono state conferite proprietà straordinarie in termini di velocità, affidabilità e possibilità di visione all'interno dei chip. La programmazione (o perlomeno il debug) è stata resa può veloce e "gratificante", senza per questo cambiare i tool utilizzati dai programmatori. A questo punto è necessario chiedersi se si tratta solamente di un esercizio puramente teorico. Ovviamente no. Esistono numerose società specializzate in questo settore. Si tratta di realtà magari poco conosciute, ma che godono di una notevole reputazione presso i loro clienti. Virtutech e VaST sono due di queste aziende che, con prodotti quali Simics, CoMET e METeor forniscono un valido ausilio per la virtualizzazione del progetto delle componenti software e hardware. Non si tratta di un nuovo modo di progettare l'hardware, ma di un metodo innovativo per sviluppare il software indipendentemente dall'hardware. Lo sviluppo del software virtualizzato, o la programmazione precedente alla disponibilità dell'hardware, non presenta richieste particolari allo staff che si occupa di programmazione e contribuisce ad alleggerire il lavoro degli addetti allo sviluppo hardware. I due gruppi diventano così più indipendenti e possono perseguire separamene i loro obiettivi. Alla fine, il prodotto risultante dall'unione di queste componenti hardware e software sviluppate separatamente sarà senza dubbio migliore e più affidabile.

(*) L'articolo è stato sviluppato in collaborazione con VaST Design Systems (www.vastsystems.com) e Virtutech (www.virtutech.com)