Dual core per soluzioni embedded

I processori basati sulla tecnologia dual-core rappresentano oggi un passo importante per il settore embedded. Vengono utilizzati in diversi campi, dalle soluzioni video a quelli che richiedono un elevato grado prestazionale

Francesco Pentella

I sistemi identificati come dual-processor sono quelli che contengono due computer, fisicamente separati, in uno stesso chassis. In un sistema del genere, questi due processori potrebbero essere localizzati sulla stessa motherboard o su di una board separata. Una tecnologia dual-core di tipo nativo è una soluzione che include due complete unità di esecuzione, chiamate core. In questa contesto possiamo trovare due processori, con le loro cache e i loro controllori, su un singolo chip.

Una soluzione del genere dispone di diversi vantaggi. Ogni core, per esempio, può avere la propria cache proprietaria e il relativo sistema operativo ha sufficienti risorse per gestire in maniera efficiente un insieme di processi in parallelo: in questo senso si ha un vero e proprio potenziamento del multitasking.

A volte questa soluzione è spesso indicata come multicore; in questo caso per multicore si intende la gestione di due processori separati (o anche quad-core, dipende dal numero di core presenti).

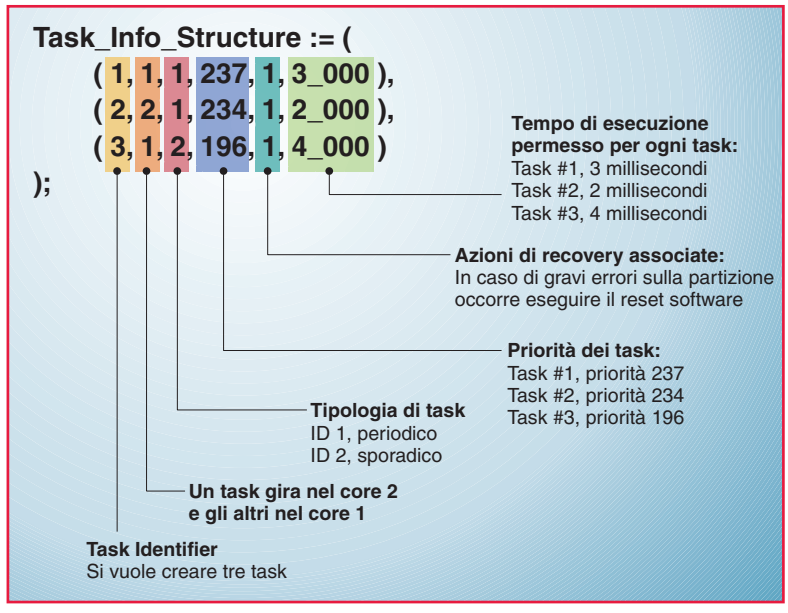

Fig. 1 – Esempio di strutture dati contenenti particolari direttive per la creazione di task presenti nel sistema

Il mercato è sempre sensibile a migliorare le proprie prestazioni per rispondere in maniera rapida alle continue sollecitazioni esterne. Questa crescita tecnologica ha avuto, come stimoli, le seguenti considerazioni:

- la richiesta continua dell'aumento della larghezza di banda per supportare l'incremento del traffico di rete, per le applicazioni orientate al networking;
- potenziare l'affidabilità del servizio. In quest'ottica si è cercato, e si cerca, di fornire dei sistemi sicuri di rete per ostacolare minacce reali e potenziali e combattere lo spamming;

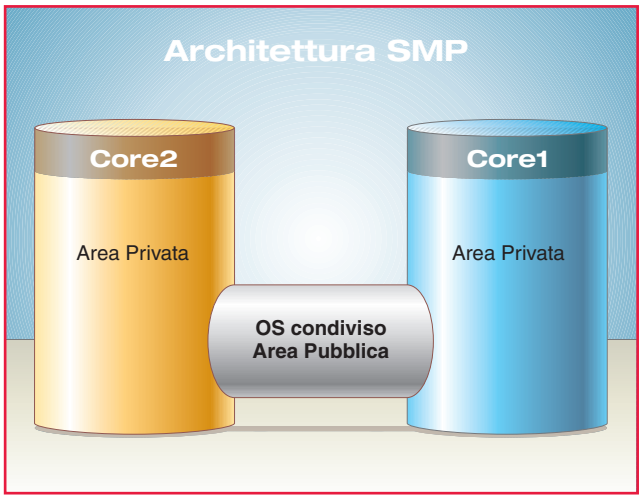

Fig. 2 – Symmetric Multi-Processing

-potenziare i servizi di mobile per venire incontro alle continue sollecitazioni commerciali.

In passato si è cercato di rispondere a queste esigenze facendo in modo di potenziare la frequenza di funzionamento dei microprocessori, ma, recentemente, questa tendenza ha subito un mutamento dovuto all'introduzione di sistemi di calcolo e computazione paralleli.

Da un punto di vista software un processore dual-core appare come un microprocessore tradizionale composto da due single core chip, un SMP (symmetric multiprocessing) dove tutte le considerazioni tecniche, come per esempio le problematiche di ottimizzazione, rimangono comunque applicabili.

Le architetture dual-core differiscono da quelle single-core SMP perché gli accessi in memoria sono per chip.

Con questo si vuole affermare che i sistemi dual-core hanno sia una bassa memory latency e sia un basso memory bandwith.

Ogni core su un dual-core AMD opteron dispone di 128 Kb di cache primaria (256 Kb/chip) e 1 Mb di secondaria (2Mb/chip): negli ultimi modelli esiste poi la cache condivisa (L3).

Se la tecnologia hardware è in continua evoluzione, il software non è da meno: i sistemi operativi per queste architetture adottano nuovi algoritmi di schedulazione.

La tecnologia di riferimento per queste soluzioni è chiamata thread-level parallelism (TLP). Il TLP è la parte del sistema operativo, o applicativo, che si prende carico di mettere in esecuzione e controllare i thread multipli in maniera simultanea: per thread si intende un'unità di programma che può essere posta in esecuzione in maniera indipendente con le altre componenti.

Virtualizzazione

Quando si parla di sistemi dual-core sempre più spesso vengono messe in evidenza le considerazioni sulla virtualizzazione: questo concetto racchiude una serie di aspetti tecnici.

Con questa tecnica si intende una particolare esecuzione di più sistemi operativi su una stessa piattaforma; l'esecuzione viene garantita dal fatto che sono emulate differenti macchine. L'impressione che se ne ricava è di avere differenti piattaforme virtuali; questo grazie all'impiego di diverse tecniche che coinvolgono risorse hardware o solamente software.

Ciascuna applicazione che gira su una piattaforma virtuale ha il controllo (virtuale) delle risorse fisiche della piattaforma stessa.

Le tecniche software utilizzate hanno il pregio di non richiedere particolari requisiti hardware (come per esempio un dispositivo MMU). Gli algoritmi utilizzati per soddisfare, dal punto di vista software, i meccanismi di virtualizzazione sono chiamate hypervisor o binary caching. Altri, invece, prevedono di introdurre algoritmi di traduzioni.

Accorgimenti software

L'applicazione messa in esecuzione in un determinato momento è analizzata alla ricerca di istruzioni particolari, quali le istruzioni privilegiate. Queste vengono sostituite con degli opcode equivalenti che devono simulare le istruzioni stesse. In questo caso il sistema operativo che ha il possesso della piattaforma non ha il reale controllo e quindi qualsiasi failure di un'applicazione ospite (con il suo sistema operativo) non pregiudica il controllo della piattaforma stessa, ma pone in uno stato di failure la partizione coinvolta.

Il termine binary caching è derivato dal processo di traduzione. Infatti, una volta fatta la traduzione dagli istruzioni reali a quelli virtuali (emulate), la conversione stessa viene salvata nella cache; in questo modo non è più necessario ripetere ogni volta la il processo di traduzione.

Questa tecnica è utilizzata in una serie di prodotti come VMWare.Un'altra tecnica utilizzata è la cosiddetta paravirtualizzazione, in cui ogni macchina virtuale ha un insieme di primitive (API) che consentono di accedere ai vari dispositivi hardware presenti nella piattaforma. In questo contesto solo i sistemi operativi riconosciuti dalla piattaforma possono essere ospitati, infatti solo questi conoscono le interfacce messe a disposizione. In questo modo, il sistema operativo ospitato è stato modificato per accedere a queste API.

Il sistema operativo della piattaforma è posto in esecuzione a ring 0, mentre i sistemi operativi ospitati in una macchina virtuale sono messi in esecuzione a ring 3 (livello di privilegio inferiore e quindi perfettamente controllabile).

Accorgimenti hardware

Dalla tecnologia dei vecchi microprocessori si conoscono i vari layer utilizzati: ring 0 o ring 3.
Il ring 0 è il livello di esecuzione del sistema operativo. Con questo livello si ha il controllo totale della macchina.
Il ring 3 è invece un livello di esecuzione a basso privilegio: di norma in questo strato girano le applicazioni utenti.
Le soluzioni hardware per la virtualizzazione sono racchiuse nel livello 0. In pratica si pongono in evidenza le estensioni proprietarie al codice macchina, la particolarità di queste estensioni è di costruire un livello di esecuzione ancora più privilegiato del ring 0. È possibile, a questo punto, identificare alcune caratteristiche e proprietà della virtualizzazione:

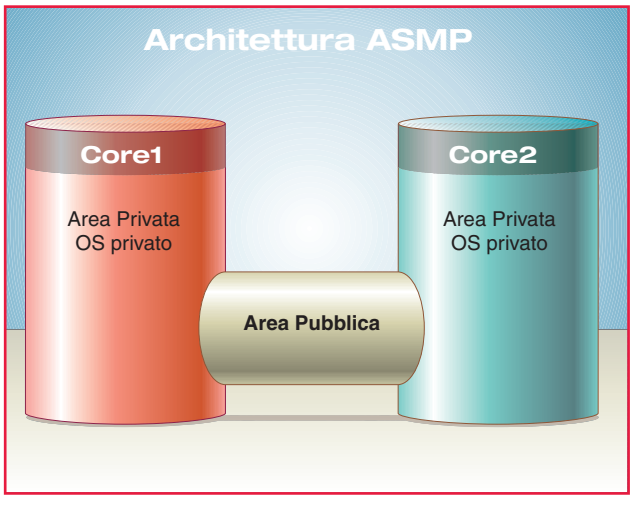

Fig. 3 – Asymmetric Multi-Processing

- Partizionamento in spazio - Ogni crash o failure in una partizione, cioè l'applicazione che è in esecuzione in una macchina virtuale, non pregiudica l'esecuzione del sistema; nel senso che la partizione in failure può essere isolata con delle opportune azioni di recovery.
- Partizionamento in tempo - Ogni segmento, partizione, è messo in esecuzione in maniera concorrente. In ogni partizione è presente un proprio sistema operativo.
- Limiti e quote - Occorre definire opportune politiche di allocazione della memoria e delle risorse fisiche del sistema per impedire usi esclusivi e protratti nel tempo di risorse o la richiesta e l'allocazione di estese quantitativi di memoria.

Partizionamento del software

La scrittura del software in un sistema del genere certamente deve ricavare vantaggi da una architettura di questo tipo.
Quando si discute di architetture del genere, allora necessariamente occorre riconoscere che esistono due grandi famiglie: sistemi di tipo simmetrici e asimmetrici, in particolar modo symmetric multi-processor (SMP) e asymmetric multi-processor (ASMP); esistono poi altre soluzioni che si basano su una combinazione più o meno ibrida di queste due famiglie.
In ogni caso, può essere ipotizzabile assumere che in una soluzione di questo tipo ogni core può avere una zona di memoria privata (nel senso che è precluso l'accesso agli altri core), e una zona di memoria chiamata pubblica dove possono risiedere strutture dati condivise e l'accesso viene regolato da meccanismi di allocazione dinamica delle risorse.
In una soluzione di tipo SMP può essere presente un kernel comune e le applicazioni che sono in esecuzione nei cores accedono così a questa immagine.
Al contrario, con una soluzione di tipo ASMP ogni core presente nel sistema ha una copia del kernel. L'immagine del kernel può essere la stessa per tutti i core o, addirittura, possono esistere differenti immagini presenti nel sistema.

Symmetric Multi-Processing

Si è scritto che in una soluzione di questo tipo esiste un solo sistema operativo che è messo in esecuzione sui due processori, ognuno di questi può, quindi, accedere all'immagine del kernel presente in memoria. In questo caso, il sistema operativo ha il compito di partizionare i task tra i due processori e potrebbe farlo in modo statico attraverso l'uso di opportuni metodi di configurazione. I metodi utilizzati possono essere diversi da una scelta progettuale all'altra. È possibile, per esempio, utilizzare opportune strutture dati contenenti particolari direttive per la creazione di task presenti nel sistema: la figura 1 ne mostra un esempio abbastanza esplicativo. L'utente, attraverso questa struttura, informa il compilatore e, di conseguenza, il processo di linking dell'esigenza di creare task partizionati.
La struttura contenuta nella figura 1 fornisce, ipoteticamente, le informazione per creare 3 task.
Il significato della struttura è il seguente:

- Con il primo campo si definisce l'identificatore del task nel sistema. In questo modo si vogliono creare tre task con degli identificatori univoci.
- Il secondo definisce la loro relazione funzionale, cioè in quale cpu ciascun task deve essere eseguito.
- Con il terzo campo si vogliono dare informazioni sul tipo di task che deve essere creato, per esempio può essere un task periodico o sporadico. In questa struttura si crea un task di tipo periodico.
- Il quarto definisce la priorità di ciascun task.
- Il quinto campo fornisce le informazioni al sistema per asso-

ciare al task le eventuali procedure di recovery. Può essere possibile, per esempio, istruire il task di pilotare un reset software della partizione a fronte di una anomalia grave della partizione stessa
- L'ultimo definisce la quantità di tempo disponibile per l'esecuzione del task.

Si può invece ipotizzare di spostare la creazione dei task del sistema nella fase dinamica; in questo modo un task è creato mediante l'uso di API messe a disposizione dal sistema operativo.
Il sistema operativo, così, crea i task, li mette in relazione alle partizioni (i due core), gestisce l'ordine di creazione e controlla le risorse condivise tra i due core.
Il controllo delle risorse condivise, in un sistema di questo tipo, assume un aspetto importante. In questo caso occorre considerare un altro fattore: il modello di programmazione utilizzato.
In questo contesto il sistema può essere dimensionato opportunamente per l'uso delle risorse condivise e nessuno può accedere direttamente alle stesse. Si introduce il concetto di proprietario e utilizzatore: il proprietario decide l'uso delle risorse, mentre il ruolo dell'utilizzatore viene attribuito a chi necessita delle stessa.
La politica di attribuzione è applicata dal proprietario che sovrintende al controllo delle risorse condivise.
Certamente il sistema va dimensionato in base a una specifica applicazione; esistono diverse applicazioni che ricavano dei vantaggi dall'utilizzo di una tecnologia multi-core. I sistemi di questo tipo possono essere switch, router, radio network controller.
È possibile, per esempio, ipotizzare di assegnare a ogni core una differente missione: dal controllo delle risorse al trattamento dei dati.
Questa ipotesi risponde a diverse considerazioni, Per esempio, in base al tipo di recovery, si può decidere di resettare la parte dei dati senza per questo incidere su quella di controllo.
In un'architettura di questo genere è possibile, per esempio, condividere zone di memoria da utilizzare come aree di scambio fra i core del sistema.
Una tabella di instradamento può così essere acceduta dal core di controllo per permettere di fare l'upstream e il downstream dei dati.

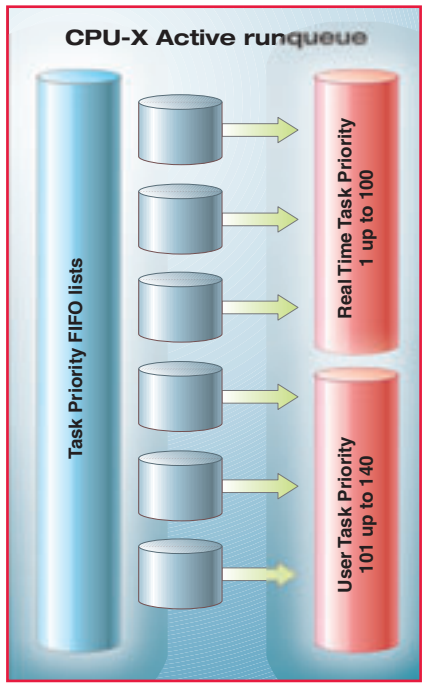

Fig. 4 – Esempio di active runqueue

Oppure, in un'altra applicazione può essere fattibile considerare la divisione delle attività del traffico della rete, cioè un core può occuparsi del traffico di upstream e l'altro di downstream.

Asymmetric Multi-Processing

In questo contesto non ci sono colloqui tra i due core. In questo modo esistono due sistemi operativi distinti ognuno con la propria immagine.
La comunicazione tra questi due mondi può avvenire attraverso una zona di memoria pubblica. Inoltre, possono esserci risorse fisiche da condividere, si pensi per esempio ad una stampante.
Particolare cura va data alla condivisione delle risorse tra i due sistemi operativi: nessun sistema è proprietario dell'intero chip; in questo modo le risorse condivise non sono, come si dice, under the hood, come in un sistema SMP. Se esiste una sola istanza di una risorsa su di un sistema multi-core, allora un core è proprietario, cioè esiste una sola istanza che riceve gli interrupt e gli eventuali messaggi di errore. In base a questo schema ci deve essere un solo attore che fornisce le risposte agli stimoli presentati e che appronti i necessari mezzi per comunicare con gli altri core. Esistono, comunque, alcuni vantaggi in un sistema ASMP.
Questo rappresenta il solo approccio quando due sistemi operativi sono inseriti in un sistema; in questo modo è possibile assegnare delle risorse dedicate a dei task critici ottenendo in cambio un sistema più deterministico. O, ancora, un sistema del genere può anche essere utilizzato per creare una pipeline di lavoro, cioè un workflow.
È un sistema che supera il grado di parallelismo non sufficientemente spinto che si ottiene con il sistema SMP. Un workflow di questo tipo potrebbe essere inefficiente quando i core condividono una cache secondaria esterna, L2. In una cache di questo tipo potrebbe succedere un disallineamento tra dati e istruzioni.

Virtual machine interface

È necessario ipotizzare alcune restrizioni quando si definisce un sistema con diverse partizioni. Ogni partizione ha una propria virtual machine, quindi a titolo di esempio è possibile prevedere i seguenti punti:

- **Memory management**

Segmentation - Il sistema di tipo guest non può definire segmenti con livelli di privilegio superiori o uguali al sistema principale.

Paging - Un sistema operativo guest ha un accesso solo in lettura alla hardware page table.

Ogni volta che un sistema guest richiede una nuova pagina, cosa che può succedere per esempio quando nel sistema guest è stato creato un nuovo processo, il sistema guest prende uno nuovo spazio di memoria che gli era stato assegnato precedentemente e comunica la richiesta al sistema principale. In questo modo il sistema guest ottiene i privilegi per utilizzare la pagina una volta che il procedimento è stato approvato dal sistema supervisore.

- **CPU**

Protection - Un sistema operativo guest deve essere messo in esecuzione a un livello di privilegio inferiore del sistema operativo principale.

Exception - Un sistema operativo guest deve avere una description table per le eccezioni con il sistema operativo principale: una fault di pagina deve avere lo stesso gestore.

System Calls - Un sistema operativo guest potrebbe avere un gestore diretto delle chiamate di sistema, in questo modo si eviterebbe di fare delle chiamate indirette al sistema operativo principale.

Interruptus - Gli eventi asincroni legati alle risorse fisiche sono sostituite da un gestore di eventi più flessibile.

Time - Ogni guest ha un proprio gestore del timer e si potrebbe anche prevedere timer di tipo virtuali.

- **Device I/O**

Network, disk - I dati sono trasferiti utilizzando canali di comunicazioni di tipo asincrono.

È cura del sistema principale esporre una serie di interfacce che permettono di accedere alle risorse hardware con una certa astrazione per soddisfare i requisiti di protezione e isolamento.

Smp e Linux

La versione 2.6 del kernel di Linux ha introdotto l'uso di un nuovo schedulatore che permette di supportare l'architettura SMP in maniera più efficiente. È in un certo senso possibile affermare che un sistema operativo ha un ruolo di mediazione. Media cioè le richieste di un programma applicativo con le risorse che dispone. Le risorse spaziano tra dispositivi fisici (stampanti e così via) e accessi in memoria.

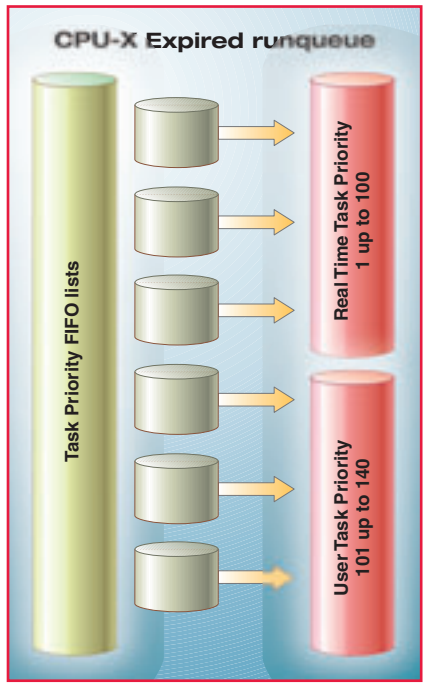

Fig. 5 – Esempio di Expired runqueue

Una componente essenziale di un sistema operativo è certamente lo schedulatore: questo assegna per uno slot temporale la risorsa CPU a una unità di programma, chiamato task. Grazie a questo schedulatore si rende possibile l'esecuzione di diversi programmi nello stesso tempo.

La versione 2.6 di Linux ha uno schedulatore progettato e implementato da Ingo Molnar.

Il pregio del lavoro svolto è stato fondamentalmente di predisporre un grado di complessità O(1).

La notazione O(n) viene utilizzata per calcolare la complessità di un algoritmo; in questo caso la complessità viene dedotta dal tempo speso per mettere in esecuzione un processo o comunque dal tempo speso per prendere una decisione in merito alla schedulazione. Il tempo per un algoritmo O(n) dipende dai suoi dati in ingresso; quando è definito come O(1) allora si può affermare che l'algoritmo è indipendente dai dati in ingresso e opera in maniera costante.

Il tempo speso dallo schedulatore non dipende dal numero dei task attivi, ma dal numero di priorità (viene utilizzata una struttura bitmap). In questo modo il costo è di O(1) per la versione 2.6 dello schedulatore, perché il tempo preso dall'algoritmo è fisso e deterministico nonostante il numero di task attivi.

Il nuovo schedulatore lavora nel modo seguente: ogni CPU ha una coda dei task in esecuzione, ordinati con il meccanismo FIFO, costituiti da 140 liste di priorità.

I task al termine della propria esecuzione sono aggiunti alla fine della lista rispettiva. Ogni task ha uno slot temporale che determina per quanto tempo al task è permessa l'esecuzione. Le prime 100 liste di priorità sono riservati per i task real-time e gli ultimi 40 sono task utenti.

La coda di attivazione è chiamata active runqueue e accanto a questa lista ne esiste un'altra: la expired runqueue.

Questa lista viene riempita dai task che esauriscono tutto lo slot a loro disposizione.

Quando vengono spostati, questo time slice, insieme alla sua priorità, viene ricalcolato.

Se non esiste alcun task nella active runqueue, per una certa priorità, allora le liste vengono scambiate. Le figure 4 e 5 mostrano schematicamente l'algoritmo utilizzato.