Macchine virtuali (*Virtual Machines*)

Se si prescinde dalle prestazioni, attraverso lo scheduling dell'unità centrale e la memoria virtuale, un S.O. può creare l'illusione che ciascun processo abbia l'uso esclusivo delle risorse. Inoltre, riprendendo il concetto di "macchina astratta", si può pensare che, dal punto di vista del processo, il S.O. rappresenti effettivamente la macchina su cui questo viene eseguito.

Portando agli estremi il concetto di stratificazione introdotto con il S.O., si ottengono le cosiddette **macchine virtuali**, il cui obiettivo è quello di ricreare ad un livello superiore una vista sia analoga a quella che si ha al livello più basso, cioè a quello fisico (hardware). Questo può essere ottenuto attraverso un S.O. semplificato, che per "virtualizzare" l'harware implementa solo un sottoinsieme dei servizi di un classico S.O. (scheduling, gestione della memoria, gestione dei dispositivi periferici), mentre non comprende altre funzionalità più avanzate (system calls, file system, ...). Un tale S.O. è detto ***Virtual Machine Monitor*** (VMM), e si occuppa soltanto della gestione delle macchine virtuali.

Poiché una macchina virtuale simula una macchina fisica, su ciascuna macchina virtuale definita dal VMM sarà possibile caricare un'istanza di un S.O. (in generale queste potranno essere anche tutte diverse una dall'altra).

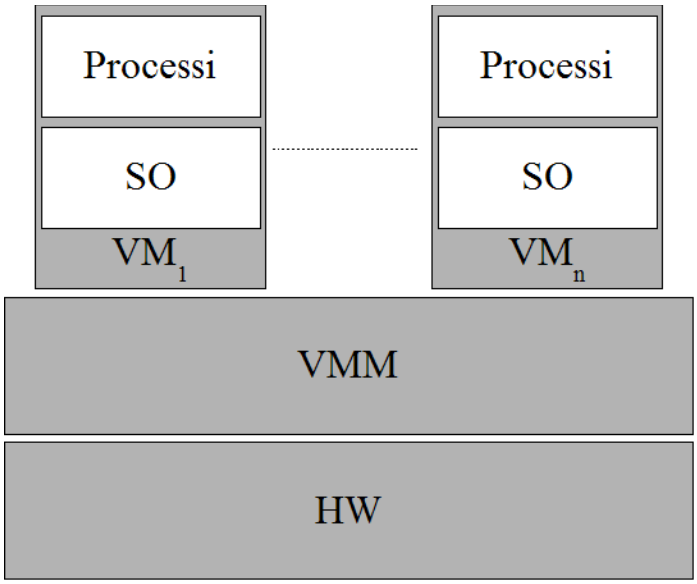

Figura 1: Il Virtual Machine Monitor gestisce le diverse macchine virtuali definite sul sistema.

Poiché deve essere garantito il corretto funzionamento dell'intero sistema, le diverse macchine virtuali dovranno operare in completo isolamento una dall'altra, lasciando al VMM le funzioni di controllo e gestione delle VM. Quindi, nel caso dell'architettura di riferimento che abbiamo considerato fino ad adesso, il VMM opererà in modo di sistema, mentre ogni VM (quindi i processi, ma anche le diverse istanze dei S.O.) dovrà essere eseguita in modo utente. Per preservare le caratteristiche dei singoli S.O. all'interno delle VM, quindi, si dovranno adottare particolari accorgimenti:

- per ogni VM si dovranno simulare i due modi di funzionamento, di sistema e utente. In particolare, il VMM terrà traccia del modo in cui si trova ogni singola VM.
- l'accesso alle risorse HW dovrà essere mediato (o simulato!) dal VMM.

Per comprendere meglio il problema, analizzeremo in dettaglio un caso particolare (ma significativo), ovvero l'invocazione di una primitiva di sistema da parte di un processo utente in esecuzione su una VM per accedere ad una periferica di I/O:

1. nel momento in cui il processo sta per invocare la primitiva di sistema, questo si trova in modo utente. L'invocazione della primitiva comporta la generazione di un'interruzione, e quindi il passaggio in modo di sistema; a questo punto il controllo passa al VMM. Poiché questo è sempre a conoscenza di quale VM è attualmente in esecuzione, provvederà ad impostare per questa il modo di sistema virtuale e quindi ad eseguire le opportune operazioni affinché il controllo passi a tale VM.
2. Il VMM passa il controllo alla VM selezionata, riportando il sistema in modo utente. A questo punto il S.O. in esecuzione sulla VM procederà nell'esecuzione della primitiva finché non incontrerà un'istruzione privilegiata per l'accesso al dispositivo periferico. Trovandosi il sistema in modo utente, l'esecuzione di una tale istruzione comporterà un'interruzione (*trap*), ed il controllo passerà al VMM.
3. Il VMM verifica anzitutto che la VM che ha generato la trap si trovi in modo di sistema virtuale (altrimenti dovrà segnalare al S.O. della VM che un suo processo ha commesso una violazione), e quindi eseguirà l'operazione per conto della VM. Nel caso più semplice eseguirà semplicemente la medesima istruzione che il S.O. ospitato avrebbe voluto eseguire, e provvederà a restituirgli il risultato; in altri casi dovrà eseguire una serie di operazioni più complesse (si pensi, ad esempio, al caso di un sistema con un solo disco fisico, e sul quale sono in esecuzione più VM, a ciascuna delle quali viene assegnata un'unità a disco: l'unità fisica sarà stata partizionata in maniera tale da assegnare una parte dello spazio ad ogni VM, ed il VMM dovrà rimappare opportunamente le richieste di lettura/scrittura da parte delle VM sulle rispettive partizioni).
4. Completata l'esecuzione dell'istruzione privilegiata, il controllo ritorna al S.O. della VM per proseguire l'esecuzione della primitiva di sistema. Il procedimento si ripete finché non si raggiunge la fine della primitiva. A questo punto la VM cercherà di eseguire un'istruzione per il ritorno dall'interruzione (es. RTI o IRET), che ancora una volta è un'istruzione privilegiata, per cui si genera nuovamente un'interruzione.
5. A questo punto, ovviamente dopo aver verificato che la VM si trovi in modo virtuale di sistema, riconosciuta la particolare istruzione, il VMM provvede ad impostare il modo utente virtuale, e quindi a restituire il controllo alla VM (al processo che aveva eseguito la system call, oppure ad un altro processo selezionato dalla VM durante l'esecuzione della suddetta system call).