Consiglio Nazionale delle Ricerche

Introduzione al VM/370

G. Cresci - M. Sommani

164

CNUCE

Servizio Sistemi di Calcolo

A cura di: Guglielmo Cresci
Marco Sommani

I N D I C E

INTRODUZIONE

Il presente manuale vuole essere una descrizione introduttiva al sistema VM/370 e come tale non fornisce informazioni sfruttabili immediatamente dal punto di vista operativo ma si limita a elencare i concetti fondamentali (la filosofia) del VM/370 cercando di chiarirne i motivi ispiratori e di evidenziare le maggiori differenze dai sistemi tradizionali, sia dal punto di vista dell'utente finale, sia dal punto di vista dell'architettura.
In quest'ottica il manuale e' stato suddiviso in tre parti che descrivono rispettivamente le caratteristiche esterne e la logica di funzionamento dei due componenti piu' importanti e di uso piu' generale: il Control Program e il Conversational Monitor System.
Le descrizioni di cui sopra si limitano a trattare criteri fondamentali come si conviene ad un testo destinato a persone del tutto digiune dell'argomento. Tutti gli argomenti trattati nel seguito possono comunque essere approfonditi sui manuali IBM che forniscono informazioni molto dettagliate e sono quindi in grado di soddisfare qualunque esigenza. Le parti relative alla descrizione dei componenti CP e CMS sono state inserite in questa trattazione introduttiva per due distinti motivi:

> per rendere disponibile a coloro che intendono intraprendere lo studio delle caratteristiche interne del sistema VM/370, uno strumento di primo approccio semplice e speriamo sufficientemente chiaro da consultare prima di affrontare l'impatto coi manuali specialistici sicuramente piu' completi ma anche piu' ostici e meno didattici;

perche' anche coloro che si apprestano a questa lettura con l'intenzione di usare il sistema solo dal punto di vista dell'utente possano conoscere le nozioni basilari della logica di funzionamento del sistema.
Questa conoscenza che ad una prima analisi puo' sembrare inutile, si rivela invece di fondamentale importanza quando, raggiunta la padronanza di quei comandi che sono strettamente necessari alle proprie elaborazioni, si voglia approfondire la conoscenza del sistema sia per usarlo per scopi diversi sia per migliorare il funzionamento e il rendimento del proprio lavoro. Su quest'ultima considerazione vorremmo insistere in modo particolare; il VM/370 e' un sistema di calcolo che consente all'utente di intraprendere elaborazioni richiedendo nozioni preliminari minime: si tratta cioe' di un sistema molto semplice. Tale semplicita' d'uso non deve pero' trarre in inganno infatti

se e' vero che si puo' fare molto con poco studio e anche vero che molto spesso approfondendo un poco la conoscenza del sistema si possono ottenere risultati molto migliori semplificando notevolmente il proprio lavoro, riducendo i tempi di elaborazioni, costruendo procedure automatiche ecc. La semplicita' del VM che consente all'utente di ottenere con poco studio iniziale il risultato di cui ha bisogno non deve indurlo, come purtroppo talvolta succede, a disinteressarsi completamente del sistema di calcolo. L'azione di aggiornamento consente infatti di sfruttare al meglio la possibilita' di calcolo, come gia' detto prima, ed anche di mantenersi sempre al passo con lo sviluppo del sistema.

PARTE PRIMA:

DESCRIZIONE ESTERNA DEL SISTEMA

Il VM/370 e' un sistema a divisione di tempo (time-sharing) per la gestione di elaboratori IBM della serie /370.
Esso e' infatti studiato per elaborare concorrenzialmente processi diversi (per es: programmi d'utente). Le risorse dell'elaboratore sono ripartite tra tutti i processi che ne fanno richiesta secondo una lista di priorita' valutata dal sistema stesso. I processi aventi priorita' piu' alta sono serviti per primi; la loro esecuzione tuttavia non puo' valicare certi limiti di tempo fissi e predeterminati, scaduti i quali essi vengono accantonati e le risorse dell'elaboratore vengono dedicate ai processi rimasti in attesa sempre seguendo il criterio di priorita' gia' accennato.
Il VM/370 mette a disposizione di ogni utente l'equivalente funzionale di un sistema di elaborazione reale: cosicche' ciascun utente ha la possibilita' di eseguire tutte le funzioni implementate su un calcolatore IBM/370 o /360. Poiche' questo equivalente funzionale e' simulato per l'utente dal VM/370 e non esiste realmente, esso e' chiamato "macchina virtuale".
Possedere una macchina virtuale equivale dunque, dal punto di vista funzionale, a possedere un elaboratore "reale", per permettere l'esecuzione di programmi in modo semplice e' pero' necessario disporre anche di un sistema operativo che si accolli i problemi connessi alla gestione dell'elaboratore.
Il VM/370 fornisce ai vari utenti anche sistemi operativi particolarmente studiati per sfruttare le caratteristiche dell'ambiente a macchine virtuali in cui operano.

SISTEMI A MACCHINE VIRTUALI

Prima di iniziare la descrizione dettagliata dei componenti e delle caratteristiche del sistema e' opportuno puntualizzare il concetto di macchina virtuale e chiarire quali prerogative e quali funzioni deve avere un sistema basato su tale filosofia.
Il concetto di macchina virtuale nasce e si sviluppa, anche storicamente, da quello di "memoria virtuale" di cui puo' considerarsi la logica estensione.
Un sistema a "memoria virtuale" puo' essere definito come un sistema di memorizzazione di informazioni in cui c'e', o ci puo' essere, una distinzione tra l'indirizzo "logico" generato da un programma e l'indirizzo "fisico", nella memoria reale, in cui l'informazione e' realmente memorizzata. Questa definizione include anche tipi di memoria molto comuni, tuttavia, il termine "memoria virtuale" si riferisce normalmente all'indirizzamento di parole di memoria da parte dell'elaboratore centrale, ed in particolare, ai sistemi in cui la traduzione e "rilocazione" degli indirizzi e' realizzata dinamicamente dall'hardware della macchina. Nella sua forma piu' generale un sistema a memoria virtuale consente la suddivisione dello spazio degli indirizzi virtuali in piu' parti, ciascuna con la sua costante di rilocazione dinamicamente sostituibile.
In questo modo e' possibile:

- Tenere in memoria solo alcune delle parti in cui e' suddiviso lo spazio degli indirizzi virtuali. Le parti restanti sono tenute su memorie "ausiliarie".

- Scambiare ogni singola parte tra la memoria centrale e quella ausiliaria, secondo quanto giudica opportuno il programma di controllo del sistema.

La tecnica della memoria virtuale offre numerosi vantaggi sia al programmatore che al sistema di calcolo.
Il maggior vantaggio che essa offre al programmatore e' quello di liberarlo da tutti i problemi connessi con la gestione della memoria. In particolare, poiche' nella memoria centrale sono presenti, in ogni istante, solo quelle parti di memoria virtuale che sono realmente utilizzate, e' possibile fornire al programmatore uno spazio degli indirizzi logici molto piu' grande di quanto non sarebbe altrimenti possibile.
In questo modo possiamo evitare di lavorare con le tecniche di sovrapposizione delle informazioni ("overlay") spesso necessarie nei sistemi a memoria convenzionale.
In verita', la sovrapposizione ha luogo anche nei sistemi a

memoria virtuale, ma e' gestita automaticamente dal sistema ed e' logicamente trasparente al programmatore.

I vantaggi piu' significativi che le tecniche di memoria virtuale offrono al sistema di calcolo sono:
- una migliore utilizzazione della memoria;
- un'accresciuta potenzialita' a disposizione della multiprogrammazione.
  Nei sistemi a memoria convenzionale, la frammentazione della memoria necessaria all'allocazione di grandi regioni contingue, provoca una cattiva utilizzazione della memoria stessa. Nei sistemi a memoria virtuale, le parti in cui viene suddivisa la memoria virtuale possono essere allocate in modo discontiguo dentro la memoria principale, dovunque ci sia spazio sufficiente.
  La riduzione della frammentazione che ne deriva, unita al fatto che, in ogni istante, risiedono in memoria solo le parti attive dello spazio degli indirizzi logici, migliora l'utilizzazione della memoria e, in definitiva, aumenta il grado di multiprogrammazione del sistema.

I sistemi a macchine virtuali possono essere definiti in modo analogo ai sistemi a memoria virtuale. Un sistema a macchina virtuale, infatti, puo' essere definito come un sistema di calcolo in cui le istruzioni lanciate da un programma per realizzare una certa operazione, possono essere diverse da quelle realmente eseguite dall'hardware della macchina. Poiche' le istruzioni, generalmente richiedono indirizzi di memoria, i sistemi a macchine virtuali includono tra le loro caratteristiche funzionali, la memoria virtuale. In questo senso, il concetto di macchina virtuale, e' una generalizzazione di quello di memoria virtuale. Nei sistemi a macchine virtuali, una macchina, quella reale (o ospitante) provvede alla simulazione funzionale di uno o piu' elaboratori: le macchine virtuali. Il sistema a macchine virtuali deve sempre realizzare la simulazione funzionale di almeno quattro componenti: console di operatore, unita' di calcolo, dispositivi di I/O e memoria (i quattro componenti base di un sistema reale).
I sistemi a macchine virtuali mettono a disposizione degli utenti facilities non riscontrabili nei sistemi di altro tipo e consentono di disporre di altre facilities in modo piu' semplice e piu' conveniente di quanto non sia possibile con sistemi convenzionali. Pur riservandoci di trattare dettagliatamente in seguito i vantaggi insiti nell'uso di sistemi di questo tipo, vogliamo gia' accennare alla possibilita' di svincolare il programmatore non solo da problemi connessi alle dimensioni della memoria reale (risultato gia' acquisito dai sistemi a memoria virtuale) ma

anche da tutti gli eventuali problemi connessi alla configurazione della macchina reale. E' infatti possibile simulare su una macchina reale delle macchine virtuali aventi configurazioni diverse da quella reale (sia piu' ridotte che piu' ampie di quella reale).
Pertanto una macchina reale di modeste dimensioni puo' realizzare un "environment" di sviluppo e di prova per un sistema destinato a macchine di grandi dimensioni.

COMPONENTI DEL SISTEMA

Dalla precedente introduzione e' immediato riconoscere che il sistema operativo VM/370 deve svolgere due funzioni fondamentali:

1) La simulazione, su di un unico elaboratore reale IBM/370, di piu' macchine virtuali.

2) La gestione delle risorse delle singole macchine virtuali.

Il VM/370 e' organizzato in quattro componenti ciascuno dei quali assolve una specifica funzione. I componenti base sono:

a) Control Program (CP): il sistema supervisore dell'elaboratore reale di cui controlla le risorse, realizza le tecniche di time-sharing, crea e gestisce le macchine virtuali. In definitiva il Control Program (CP) assolve la funzione di simulazione delle macchine virtuali sulla macchina reale.

b) Conversational Monitor System (CMS): e' un sotto-sistema (perche' puo' funzionare soltanto sotto CP) di gestione di una macchina virtuale. E' mono-utente e conversazionale, permette la creazione e la gestione di files, la compilazione, messa a punto, esecuzione di programmi. Dispone inoltre di una vasta gamma di funzioni di controllo e utilita'. Il CMS e' il sistema supervisore che il VM/370 fornisce all'utente di macchina virtuale.

CP e CMS sono i componenti base del VM/370. Essi erano presenti gia' nella prima versione di VM/370 rilasciata nel 72 e forniscono facilities di uso assolutamente generale. Esistono tuttavia altri componenti che consentono la gestione di problemi particolari; essi sono:

c) Remote Spooling Communications Sub-system (RSCS): e' un sotto-sistema che, operando in macchina virtuale, consente la gestione di stazioni remote della rete TP.

d) Interactive Problem Control System (IPCS): sotto-sistema per la classificazione e l'analisi dei malfunzionamenti software del VM/370.

E' immediato riconoscere che i componenti di interesse generale sono soltanto i primi due (CP e CMS) essendo l'uso

dell'IPCS riservato ai soli responsabili della manutenzione software del sistema e quello di RSCS ai gestori delle stazioni remote dell'elaboratore.
Questo fascicolo intende occuparsi soltanto delle possibilita' e delle caratteristiche dei componenti CP e CMS.
Dei componenti del VM/370 l'unico in grado di operare su un'elaboratore reale e' il CP; gli altri, che abbiamo chiamato sotto-sistemi (CMS, RSCS, IPCS), necessitano sempre, per funzionare correttamente, della presenza del Control Program. Peraltro la disponibilita' del solo Control Program non consente un facile uso della macchina virtuale per creare files, compilare o eseguire programmi etc. perche' disporre del Control Program e di una macchina virtuale equivale a disporre soltanto di un elaboratore reale di caratteristiche uguali a quelle con cui la macchina virtuale e' stata definita.
E' infatti ancora necessario disporre di un sistema operativo che si accolli i problemi di gestione della macchina. A questo scopo il VM fornisce un componente particolare: il CMS che, come gia' accennato, sfrutta appieno le caratteristiche dell'ambiente a macchine virtuali in cui si lavora.
Oltre al CMS e' pero' possibile adoperare in macchina virtuale un qualunque sistema operativo in grado di lavorare su un elaboratore IBM/370 (vedi fig. 1).

DOS
DOS/VS
OS/PCP
OS/MFT
OS/MVT
OS/VS1
OS/VS2
OS/ASP
PS44
TSO dell'OS

Fig. 1: Sistemi operativi usabili su macchina virtuale

CARATTERISTICHE DEL SISTEMA

Il VM/370 rilasciato ufficialmente nel 1972 e' il risultato degli studi e delle sperimentazioni compiute dalla IBM nel settore dei sistemi a macchine virtuali. Esso infatti raccoglie l'eredita' di precedenti sistemi di cui conserva tuttora le caratteristiche di struttura. Il piu' noto di questi fu il CP-67, un sistema sperimentale studiato e messo a punto dal Cambridge Scientific Center destinato a calcolatori IBM/360 modello 67 che godette una certa diffusione alla fine degli anni 60. Il CP-67 svolse tutto il lavoro conversazionale del CNUCE fino al 75, anno in cui fu sostituito dal VM/370.
Rispetto a tale sistema, gia' articolato nei due componenti CP e CMS, il VM ha subito profondi cambiamenti, dettati anche dallo sviluppo tecnologico; esso e' passato, in 4 anni di vita, attraverso 5 releases (gia' si sta annunciando il sesto) che, oltre a tenerlo al passo con le novita' hardware e dotarlo di nuove possibilita' lo hanno portato ad un grado di affidabilita' notevolmente superiore a quella dei sistemi che lo precedettero e a quella delle prime versioni del VM stesso.
Le caratteristiche esterne del VM/370, che sono quelle di maggior interesse per l'utente, possono essere sintetizzate in una serie di punti (elencati di seguito) alcuni dei quali sono proprieta' esclusiva dei sistemi a macchine virtuali, altri si riferiscono a facilities riscontrabili anche su altri sistemi che pero' in ambiente VM presentano aspetti originali e inediti.

1) Versatilita'

La versalita' del VM/370 e' giustificata da diverse considerazioni che illustrano come l'uso di un sistema a macchine virtuali svincoli il programmatore da moltissimi problemi inerenti ai limiti imposti alla sua attivita' dalla configurazione hardware e software della macchina.
Si e' gia' accennato, in sede introduttiva, alla possibilita' di usare in macchina virtuale software diversi da quelli esplicitamente previsti dal VM/370 (CMS, RSCS, IPCS). La fig. 1 fornisce un'elenco che raccoglie praticamente tutti i piu' comuni sistemi operativi IBM. Questa possibilita', unita a quella di avere piu' macchine virtuali contemporaneamente attive sulla stessa macchina reale, porta alla capacita' del VM di "girare" contemporaneamente software diversi sullo stesso hardware.
A parte i limiti insiti nel fatto che la "contemporaneita'" va intesa in termini di time-sharing,

(vedi al riguardo il paragrafo "Restrizioni al lavoro in macchina virtuale") questa possibilita' consente svariate applicazioni ottenibili solo su sistemi a macchine virtuali:

- consente infatti di svolgere attivita' di programmazione (in macchine virtuali CMS) in parallelo all'attivita' di produzione vera e propria svolta con macchine virtuali DOS e/o OS.

- consente la messa a punto e la prova di nuove versioni del sistema operativo di produzione senza fermare il vecchio.

- in larga misura consente di attuare la manutenzione ed i test su unita' non vitali del calcolatore in parallelo all'attivita' di servizio. I programmi di test possono infatti "girare" in macchina virtuale.

Quelli elencati sono solo alcuni esempi di applicazioni comuni che sfruttano la possibilita' del VM di "girare" software diversi sulla stessa macchina reale, si possono pero' trovare molte altre applicazioni simili che non stiamo ad elencare.
Un'altra possibilita' unica offerta dai sistemi a macchine virtuali deriva dal fatto che la sola limitazione imposta alla configurazione di una macchina virtuale e' quella di lavorare sulla CPU della macchina reale (vedi ancora le restrizioni al lavoro in macchina virtuale). Per quanto riguarda le altre caratteristiche (taglia di memoria, numero e tipo dei dispositivi di I/O) non ci sono vincoli imposti dalla configurazione reale. E' quindi possibile simulare in macchina virtuale hardware diversi da quello reale. Anche questa capacita' rende possibili svariate applicazioni quali:

- sviluppare su macchine reali di modeste dimensioni, sistemi operativi o programmi in genere destinati a lavorare su macchine piu' grandi.

- permettere ad un sistema piu' piccolo di sostituire il sistema piu' grande usato per la produzione (per es. in caso di guasti che richiedano il fermo di quest'ultimo). Naturalmente in questa circostanza e' necessario considerare un'inevitabile degradazione delle prestazioni.

Naturalmente le possibilita' citate non sono mutuamente esclusive per cui le applicazioni esemplificate possono combinarsi tra loro in qualunque modo si voglia.

Un'ultima considerazione che gioca a favore della versatilita' del VM e' la possibilita' che il sistema fornisce, tramite opzioni e comandi di operatore, di adattarsi alle esigenze dell'utente.
Sono possibili diverse azioni in questo senso, tutte miranti a sintonizzare il sistema sul lavoro dell'utenza per esempio per favorire l'esecuzione di una macchina virtuale rispetto alle altre o per penalizzare in modo piu' o meno marcato l'uso di determinate risorse.
L'uso appropriato di questa possibilita' consente:

- di migliorare notevolmente le prestazioni ed il grado di utilizzazione del sistema.

- di ottenere buone prestazioni in condizioni di carico anche molto diverse.

- di modificare dinamicamente i parametri di sistema per eseguire le variazioni delle esigenze dell'utenza.

2) Sicurezza

I problemi della sicurezza del lavoro e delle informazioni dell'utenza sono stati affrontati, in VM, con una cura particolare che, unita alle possibilita' messe a disposizione dalla filosofia delle macchine virtuali, ha consentito di raggiungere, da questo punto di vista, risultati non riscontrabili su sistemi di altro tipo.
Innanzitutto e' rigorosamene osservato il principio di indipendenza delle macchine virtuali per cui ogni macchina virtuale:

- dispone di uno spazio disco accessibile a lei sola e quindi protetto nei confronti delle altre macchine virtuali.

- dispone di un suo sistema operativo distinto da quello delle altre macchine per cui e' automaticamente protetta da tutti i malfunzionamenti che possono avvenire nel software di altre macchine.

- la condivisibilita' di informazioni (siano esse dati, programmi o lo stesso sistema operativo), tra macchine diverse é' ammessa solo previa autorizzazione delle macchine interessate.

Queste possibilita' garantiscono la sicurezza delle informazioni contenute nella macchina virtuale. L'accesso a tali macchine e' a sua volta soggetto ad un meccanismo

di protezione: ad ogni utente e' assegnata una macchina virtuale, ogni macchina e' accessibile solo dopo aver fornito al sistema una "parola chiave" nota solamente al proprietario della macchina. Solo se il VM riconosce la parola chiave (password) consente l'accesso alla macchina virtuale, in caso contrario lo rifiuta. Si evita cosi' il pericolo che persone non autorizzate possano accedere alle altrui informazioni. A questo si aggiunga infine che l'utente ha la possibilita' di modificare dinamicamente (tramite comando da terminale) la password della propria macchina virtuale.

3) Interattivita'

Questa caratteristica e' uno dei "punti di forza" del VM pur non essendo una prerogativa dei sistemi a macchine virtuali. L'interattivita' del VM si presenta in due aspetti:

- Interattivita' nel componente CP

Il Control Program consente di gestire un notevole numero di macchine virtuali contemporaneamente. Dal punto di vista teorico non esistono limitazioni al numero di utenti che possano accedere contemporaneamente ad un sistema VM, in pratica una volta che siano saturate tutte le risorse a disposizione, ogni nuovo utente comporta un aggravio del carico di sistema che si manifesta in una certa degradazione delle prestazioni. Tuttavia, poiche' la saturazione delle risorse si ha generalmente con numeri di utenti collegati notevolmente elevati, e' corretto affermare che l'interattivita' del componente CP e' assai spinta. Questa valutazione e' confermata da numerosi studi di confronto tra le prestazioni dei sistemi multi-accesso piu' diffusi, tutti concordi nel riservare al VM/370 una posizione di preminenza tanto piu' marcata quanto piu' elevato e' il numero di utenti collegato.

- Interattivita' nel componente CMS

L'interattivita' del componente CMS e' da intendere come l'insieme delle possibilita' offerte al programmatore per interagire col sistema in tutte le fasi del lavoro (creazione di files, compilazione, esecuzione di programma). In questo senso e' corretto affermare che l'interattivita' del VM e' molto spinta anche nel componente CMS che dispone tra l'altro di un editore e di un insieme di facilities di messa a punto dei programmi veramente completo.

L'insieme di queste due possibilita' fa si che i tempi necessari alla scrittura e alla messa a punto di un programma in ambiente CMS siano di gran lunga inferiori (spesso si riducono di 1 ordine di grandezza) di quelli richiesti da un sistema conversazionale di altro tipo. L'ambiente CMS risulta pertanto il piu' adatto all'attivita' di programmazione poiche' riesce ad accrescerne notevolmente la produttivita'.

4) Semplicita'

Altro punto a favore del sistema VM e' la semplicita' di uso della macchina virtuale. Questa caratteristica e' spesso "propagandata" anche per diffondere l'uso del sistema e convincere anche i piu' "tradizionalisti" a battere la strada VM. Tuttavia, nonostante l'abuso che talvolta ne e' fatto, e' pur sempre vero che la semplicita' con cui e' possibile cominciare ad eseguire lavori di programmazione in ambiente CMS e' un fatto universalmente riconosciuto. La possibilita' di poter lavorare senza avere nozioni relative alla struttura ed al tipo di funzionamento del sistema ma solo conoscendo pochi e semplici comandi, fa' del VM/CMS un ambiente molto utile per scopi didattici sia per introdurre gli allievi allo studio di sistemi piu' sofisticati, sia per tenere lezioni e consentire esercitazioni pratiche su particolari linguaggi.
Al di la' delle implicazioni didattiche che la semplicita' del sistema puo' avere, questa caratteristica talvolta contribuisce a sopire nell'utenza VM la tendenza ad approfondire le proprie conoscenze. Questo atteggiamento che purtroppo ha una certa diffusione, e' assolutamente negativo poiche' il discorso della semplicita' d'uso e' vero finche' ci si riferisce al periodo di iniziazione (al "rodaggio") del nuovo utente. Se questo pero' desidera lavorare abitualmente in ambiente VM, e' bene che continui ad approfondire la conoscenza del sistema perche' i risultati del suo lavoro ne avranno sicuramente grandi benefici. In conclusione e' molto facile imparare ad usare la macchina virtuale, non e' altrettanto facile saperne sfruttare appieno le possibilita': per far questo e' necessario approfondire lo studio della struttura e del funzionamento del sistema.

IL SOFTWARE DELLA MACCHINA VIRTUALE

Gia' si e' accennato, parlando di componenti del sistema, ai compiti del Control Program e del Conversational Monitor System, vogliamo ora generalizzare il discorso delle possibilita' di lavoro offerte dalla macchina virtuale con particolare riguardo alla disponibilita' di software ed ai possibili modi di impiego.
Il Control Program fornisce soltanto il supporto di gestione delle macchine virtuali; le risorse della singola macchina virtuale devono invece essere affidate ad un altro sistema operativo che puo' essere uno di quelli rilasciati tra i componenti del VM (per esempio il CMS) oppure un altro sistema operativo in grado di gestire un calcolatore IBM/370 (vedi al riguardo la fig. 1). Questi sistemi operativi lavorano come se disponessero di un calcolatore reale (completo di CPU, memoria, I/O, console), il CP si incarica poi di interfacciare per loro le risorse reali con quelle virtuali (cioe' definite alla macchina virtuale e quindi trattate dal sistema operativo della macchina virtuale come reali).

Si noti quindi come in ambiente VM ci sia un componente (il CP) che sta al di sopra di tutti gli altri sistemi operativi: un Supervisore dei Supervisori.
Per questa sua caratteristica il Control Program viene spesso definito "Sistema Ipervisore".

Le applicazioni possibili in Macchina Virtuale sono tutte quelle rese disponibili dai sistemi operativi che vi possono lavorare. Schematicamente sono possibili:

- Applicazioni di tipo batch (per esempio con sistemi operativi DOS/VS o OS/VS).
- Applicazioni interattive a utente singolo (CMS)
- Applicazioni multi-accesso (con CICS-VS, VM/370 o TSO-OS)

Inoltre sono possibili tutte le combinazioni delle precedenti applicazioni ed e' proprio in questo tipo di attivita' che l'uso del VM/370 diviene conveniente. Si tratta infatti dell'unico sistema di calcolo che consente di usare contemporaneamente software diversi e quindi di svolgere in parallelo attivita' diverse. Infatti se si usa un solo tipo di applicazione, per es. batch per produzione, ha poco senso adoperare il VM con un sistema batch in macchina virtuale, e' sicuramente piu' conveniente usare quel sistema sulla macchina reale.
Se pero' insieme all'attivita' di produzione, si vuole

svolgere anche un'attivita' di sviluppo di programmi (creazione e/o aggiornamento di programmi) l'uso del VM diviene molto conveniente perche' consente di svolgere quest'ultima attivita' in parallelo alla prima organizzando il sistema in una macchina virtuale di produzione batch e tante macchine virtuali CMS per sviluppo di programmi. Una situazione di questo tipo e' esemplificata nella figura n. 2.

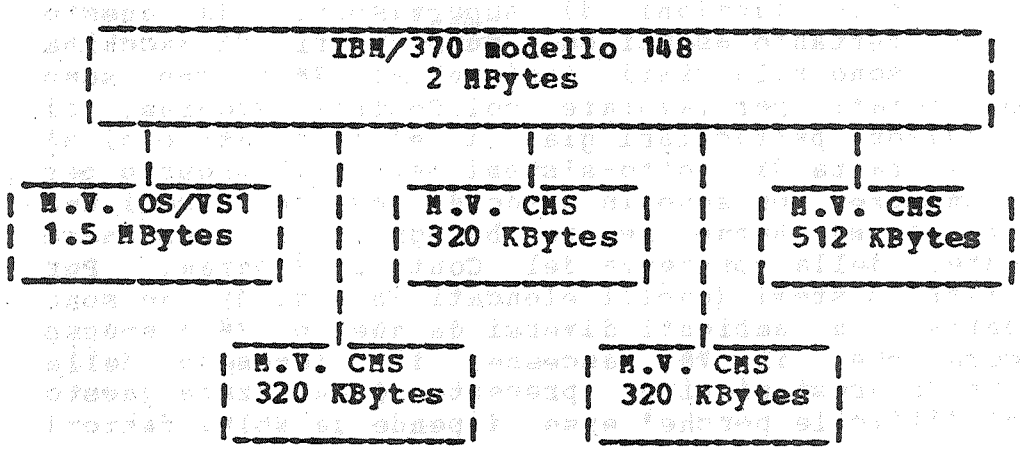


Fig. 2

dove si vede come su un hardware IBM/370 modello 148 con 2MB di memoria reale possano convivere una macchina virtuale OS/VS1 per la produzione batch con memoria virtuale di 1.5 MBytes e 4 macchine virtuali CMS aventi memoria virtuale di 320K e 512K. Notiamo ancora che questa soluzione non sarebbe possibile con altri sistemi di calcolo; l'unica soluzione alternativa a questa sarebbe quella di lavorare con il sistema di produzione in determinate ore del giorno e con un altro sistema per il lavoro di sviluppo dei programmi in altro orario. Come e' facile immaginare questa alternativa e' notevolmente piu' scomoda e macchinosa della precedente poiche' comporta frequenti operazioni manuali e fermi del servizio.
Come si e' visto non ci sono preclusioni all'uso sotto VM di qualunque sistema operativo disegnato per gestire un calcolatore IBM/370, comprendendo tra questi tutti quelli gia' elencati in fig. 1 e lo stesso VM/370. In pratica pero' lavorare in macchina virtuale significa avere tra i programmi di utente e l'hardware della macchina che li deve eseguire non piu' uno ma due intermediari: il sistema operativo della macchina virtuale ed il sistema operativo

della macchina reale (il Control Program). Questo significa che tutte le richieste che il programma d'utente invia al supervisore devono essere gestite da 2 supervisori e quindi c'e' un dispendio di tempo maggiore di quanto non sarebbe necessario se lo stesso programma girasse sullo stesso hardware sotto il controllo di un solo supervisore (quello che in VM e' il supervisore della macchina virtuale). In pratica quindi lavorare in macchina virtuale implica una certa degradazione di prestazioni del sistema.
Questa degradazione dipende essenzialmente dalla duplicazione delle funzioni di supervisore: da questo fenomeno sono pertanto esenti quei supervisori di macchina virtuale che sono rilasciati insieme al VM e che sono pertanto progettati per lavorare col Control Program. Di questi supervisori particolari gia' si e' accennato e si e' detto che si tratta di sotto-sistemi perche', proprio per evitare ridondanze, non sono in grado di gestire da soli una macchina reale, ma hanno sempre bisogno, per funzionare correttamente, della presenza del Control Program. Per tutti gli altri sistemi (quelli elencati in fig. 1) che sono stati progettati in ambienti diversi da quello VM e spesso prima ancora che il VM nascesse, il fenomeno della degradazione di prestazioni e' presente. Quantizzare questo fenomeno e' difficile perche' esso dipende da molti fattori quali:

- il tipo di lavoro che si vuole eseguire (lavori che implicano minori interventi del supervisore risultano meno degradati)
- il numero ed il tipo delle macchine virtuali che concorrono all'utilizzo delle risorse del sistema (maggiore e' il numero delle macchine virtuali attive nel sistema, maggiore e' la degradazione di prestazioni).

Esistono tuttavia dei mezzi per limitare questa degradazione che sono sorti in tempi molto posteriori alla nascita del VM e sono stati sviluppati solo quando il sistema VM/370 ha raggiunto una notevole diffusione.
Questa problematica rappresenta uno dei settori della ricerca hardware e software attualmente in piu' rapida evoluzione. Allo stato attuale il problema ha avuto due diversi approcci:

- Uno di tipo hardware che cerca di realizzare, via microprogrammi, un notevole numero di funzioni per evitare il piu' possibile alle macchine virtuali il ricorso al Control Program. Si hanno cosi' notevoli risparmi di tempo per es. nella gestione di operazioni privilegiate, nelle operazioni di I/O, etc... Questi studi hanno portato a tutt'oggi a due dispositivi hardware diversi:

1) VMA (Virtual Machine Assist feature) installabile su tutti i modelli della serie IBM/370.

2) ECPS (Extended Control Program Support) disponibile solo sui modelli 138 e 148 e molto piu' potente e vantaggioso dell'altra.

- Uno di tipo software che cerca di fare interagire di piu' i supervisori delle macchine virtuali con il Control Program per evitare al massimo la ridondanza e la duplicazione di funzioni. L'insieme di queste tecniche software va sotto il nome di "handshaking" ed ha portato ad ottimi risultati per i sistemi DOS/VS e OS/VS1.

RESTRIZIONI AL FUNZIONAMENTO IN MACCHINA VIRTUALE

Tutto quanto detto finora relativamente alla possibilita' di lavoro in macchina virtule e' vero se non vengono violate alcune regole che limitano tali possibilita'.
Si tratta di condizioni che sono ampliamente verificate nella maggioranza dei casi e che si applicano solo a situazioni molto particolari. Le piu' importanti sono:

- Non e' possibile lavorare correttamente in macchina virtuale se si usano programmi che dipendono strettamente dal tempo. Il funzionamento del VM e' infatti basato sull'assegnazione a ciascuna macchina virtuale di piccoli intervalli di tempo scaduti i quali la macchina viene estromessa dall'unita' di calcolo e richiamata solo dopo un tempo che e' variabile a seconda del carico del sistema.

- Non si ha garanzia di buon funzionamento in macchina virtuale se si lavora con programmi che implicano funzioni dipendenti dal modello di elaboratore. Poiche' il VM e' destinato a tutti gli elaboratori della serie /370 le funzioni dipendenti dal modello dell'elaboratore non sono eseguite correttamente in macchina virtuale.

- Alcuni limiti esistono pure sulle possibilita' di usare, in macchina virtuale, programmi di canale "automodificantesi", cioe' programmi di canale che vengono alterati nell'intervallo di tempo che intercorre tra l'inizio ed il termine della operazione di I/O. In genere si dice che il VM non supporta questo tipo di operazioni a meno che non si tratti di programmi generati dal metodo d'accesso ISAM dell'OS o dal metodo d'accesso TCAM livello 5 dell'OS/VS. La limitazione all'uso dei programmi di canale "automodificantesi" cade sempre se la macchina virtuale che li esegue e' definita nella regione V=R del sistema.

PARTE SECONDA:

DESCRIZIONE DEL COMPONENTE CP

FILOSOFIA DEL SISTEMA VM/370

Nei capitoli precedenti si e' accennato ad alcuni dei principi informativi del sistema di calcolo VM/370. In questo vogliamo raccogliere tutti questi principi in modo da fornire un quadro completo e definire pertanto la "filosofia" di funzionamento.
Il Control Program fornisce a ciascuno dei suoi utenti una copia architettonicamente completa di un Sistema/370 che viene comunemente chiamata macchina virtuale. L'esecuzione concorrente di piu' macchine virtuali e' realizzata con tecniche di multiprogrammazione, time-sharing e gestione di memorie virtuali.
La differenza fondamentale tra programmi d'utente che si eseguono in macchina virtuale sotto VM/370 e quelli che si eseguono sotto il controllo di un sistema operativo convenzionale e' che i primi hanno accesso a istruzioni privilegiate e possono eseguire funzioni normalmente riservate ai Supervisori. Il VM/370, come supervisore di supervisori, appartiene a quella classe di programmi noti come Ipervisori. Prima di esaminare le caratteristiche del CP che gli consentono di svolgere le funzioni di Ipervisore e di suddivisione delle risorse tra i vari utenti concorrenti, e' bene chiarire quali siano quelle particolarita' di architettura della serie /370 che consentono all'Ipervisore delle macchine virtuali di essere un sistema pratico ed efficiente.
Coloro che hanno studiato i principi che consentono ad un architettura di essere "virtualizzabile" (cioe' simulabile su macchina virtuale) sono concordi nell'affermare che presupposto fondamentale e' che l'esecuzione di alcune particolari istruzioni sia consentita solo quando il sistema e' in uno stato "privilegiato". In questi termini possono essere considerate istruzioni "particolari" tutte quelle che possono alterare o indagare lo stato dell'unita' di calcolo e dei dispositivi di I/O del sistema. L'esame dell'architettura dei sistemi /370 rivela che tutte le istruzioni di questo tipo possono essere eseguite soltanto da programmi in Stato Supervisore e quindi che ad un programma non e' permesso conoscere lo stato del Sistema in cui sta eseguendo senza fare ricorso a istruzioni privilegiate.
Altri 2 aspetti dei Sistemi /370 li rendono particolarmente adatti a realizzare architetture virtuali, si tratta:

- del vasto repertorio di istruzioni di cui questi sistemi dispongono con particolare riguardo al notevole numero di istruzioni privilegiate
- e soprattutto la capacita' di supportare memorie virtuali.

Quest'ultima caratteristica e' particolarmente importante per un sistema Ipervisore poiche' semplifica notevolmente il problema della protezione di locazioni di memoria e quello della gestione della memoria centrale.

Definite le caratteristiche di un'architettura "virtualizzabile", vediamo di entrare nel merito dei compiti fondamentali di un programma di controllo di un'architettura "virtuale".
Il Control Program del VM/370 e' stato progettato per fornire ai suoi utenti tre servizi distinti:

1) Deve presentare una pratica ed efficiente interfaccia verso le macchine virtuali e tutti i programmi che eseguono sotto il suo controllo. Questo significa che, a parte considerazioni relative ai tempi d'esecuzione, tali programmi devono eseguire in una macchina virtuale esattamente nello stesso modo in cui sarebbero stati eseguiti su un sistema /370 reale.

2) Deve fornire agli utenti una semplice interfaccia verso il sistema in modo che questi possano controllare semplicemente la macchina virtuale e la sessione di lavoro col solo ausilio di un terminale remoto.

3) Poiche' si tratta di un sistema time-sharing multi-utente, il Control Program deve allocare le risorse del sistema di calcolo reale alle macchine virtuali in modo imparziale ed efficiente.

Poiche' alcune di queste caratteristiche sono tipiche del VM/370 e lo differenziano dai sistemi tradizionali, vale la pena di discuterle piu' da vicino prima di affrontare l'esame dettagliato delle funzioni del Control Program. Tre punti risultano essere rilevanti nel confronto tra il Control Program ed i sistemi convenzionali tipo DOS, OS, etc.

1) Il Control Program ha uno scheletro ben definito ed architettonicamente stabile. L'interfaccia base visibile alle macchine virtuali e' quella definita dal "System/370 Principles of Operation". Confrontata con le interfacce fornite da altri packages software questa ha il vantaggio di essere stabile, precisa e concisamente documentata. Il VM/370 fornisce anche un'estensione a questa interfaccia consentendo chiamate dirette all'Ipervisore tramite l'istruzione "Diagnose", ma questa estensione ha solo funzioni di convenienza e di miglioramento di performance, essa non e' essenziale per il supporto del software standard.

2) Poiche' il CP fornisce soltanto "features" a livello di macchina "invisibili" ai programmi che eseguono in macchina virtuale, il suo codice risiede in una memoria virtuale separata dalla memoria usata dalle macchine virtuali che controlla. Questo sistema semplice (e quindi affidabile) garantisce la protezione del codice di sistema da tentativi di distruzione da parte di programmi d'utente.

3) Infine il Control Program non e' in alcun modo orientato verso i programmi d'utente ma solo verso l'ottimizzazione delle risorse reali. Questo fatto ha due implicazioni:

   - tutte le applicazioni software (tipo editori, compilatori, loaders) sono fornite dal CMS, un monitor conversazionale mono-utente che lavora dentro la macchina virtuale. L'isolamento dei programmi d'utente dentro le macchine virtuali e' un'ulteriore garanzia di sicurezza ed integrita'.

   - le unita' di lavoro sono definite dal CP in termini di eventi fisici (interrupts) piuttosto che tramite delimitatori logici.

Altre caratteristiche, pur non essendo esclusive dei sistemi Ipervisori time-sharing (come lo erano le precedenti), rivestono tuttavia particolare importanza; si tratta in breve:

1) della modularita' di disegno del Control Program. Un modulo fornisce sempre un solo servizio, o al piu' una serie di servizi strettamente correlati tra loro. Viceversa tutte le funzioni connesse ad un singolo servizio sono generalmente raccolte in un unico modulo.

2) delle convenzioni che regolano gli agganci tra moduli diversi. Queste sono state studiate in modo da rendere il piu' possibile semplice l'implementazione di programmi rientranti che sono un attributo essenziale per i sistemi multi-tasking.

3) della linearita' e semplicita' delle soluzioni adottate. I progettisti hanno infatti sempre tentato di adottare la via piu' semplice e diretta per risolvere i problemi. Questo tipo di approccio, se talvolta ha generato degli errori dovuti a semplificazioni eccessive, in generale ha consentito soluzioni di avanguardia per semplicita', velocita' ed affidabilita'.

LE FUNZIONI DEL CONTROL PROGRAM

Come gia' abbiamo detto, il Control Program e' costituito di tre componenti logicamente distinte che forniscono ciascuno un gruppo di servizi: essi sono l'"Ipervisore" che fornisce l'interfaccia per le macchine virtuali, il gestore delle risorse e l'Interfaccia per l'utente.
Compito principale dell'Ipervisore e' la simulazione delle istruzioni privilegiate. Poiche' il Control Program e' l'unico programma che puo' eseguire in stato di Supervisore nella macchina reale, e' evidente che tutte le istruzioni privilegiate lanciate dalle macchine virtuali (quindi in stato di problema) debbano essere intercettate dal Control Program, simulate e riflesse alla macchina virtuale. Oltre a cio' il Control Program deve fornire meccanismi software per la descrizione di tutti i dispositivi delle macchine virtuali cioe' registri dell'unita' di calcolo e canali, control units e dispositivi di I/O, e' cioe' necessaria, in aggiunta alla simulazione delle istruzioni, anche una simulazione di dispositivi di I/O che viene realizzata con tecniche diverse a seconda del dispositivo in oggetto. In relazione a queste tecniche e' possibile la seguente classificazione delle unita' di I/O:

1) Unita' di I/O dedicate

2) Unita' di I/O parzialmente simulate

3) Unita' di I/O completamente simulate

1) Le unita' di I/O dedicate sono quelle unita' virtuali che hanno un corrispondente reale ad esse perfettamente equivalente. Esse sono possedute esclusivamente dalla macchina virtuale (nastri, dischi dedicati) e la loro gestione e' affidata al sistema operativo che controlla la macchina virtuale e viene quindi ignorata dal CP. Questa gestione avviene sempre in modo sincrono con l'esecuzione della funzione richiesta.

2) Appartengono alla categoria delle unita' di I/O parzialmente simulate quelle unita' proprie di un utente che non trovano un corrispondente diretto nella configurazione reale. In questo caso tali unita' sono sempre gestite dal sistema operativo della macchina virtuale pero' le operazioni di I/O ad esse dirette vengono sempre intercettate ed opportunamente tradotte dal Control Program; esempio tipico di simili unita' sono i "minidischi".

Questi dispositivi, consistono in unita' a disco virtuali di estensione inferiore a quella dell'unita' reale; in questo modo e' possibile far risiedere fisicamente sullo stesso disco reale piu' minidischi. Questa soluzione e' giustificata dal fatto che, generalmente, la capienza di un disk-pack fisico e' molto superiore alle esigenze di spazio disco di un solo utente. Pertanto sarebbe anti-economico dedicare ad ogni utente un'intero disco reale; e' sicuramente piu' pratica la soluzione adottata di definire dischi virtuali di estensione adatta alle necessita' della macchina virtuale e di allocare piu' dischi virtuali (minidischi) su un unico dispositivo reale. Ogni minidisco conserva tutte le caratteristiche di un disco reale; esso e' infatti identificato da un indirizzo (virtuale) e da un'etichetta come il dispositivo reale. La fig. 3 schematizza una possibile suddivisione in minidischi del disco fisico avente indirizzo <u>reale</u> 230 ed etichetta VMDSK1.

VMDSK1 (230)

Ind. virt.	Etichette	Macchina virtuale proprietaria
191	MDSK1	USERA
191	CMSDSK	USERB
193	MDSK2	USERA
195	MDSK3	USERA
191	PIPPO	USERC

Fig. 3: Allocazione di minidischi su un disco reale

3) Le unita' di I/O completamente simulate dal CP sono i lettori, i perforatori e le stampanti; cioe' tutti i dispositivi di I/O lenti (detti anche unita' di tipo U/R). Per mantenere l'equivalenza funzionale col sistema reale, ogni macchina virtuale e' provvista di dispositivi di ingresso-uscita di tipo U/R.
Soprattutto per ragioni di costo, non e' conveniente dedicare una stampante od un lettore reale ad una macchina virtuale (anche se il CP lo permette), ma si preferisce simulare il funzionamento di tali unita', disponendo in realta' di un numero limitato di unita' reali.
La realizzazione di questa simulazione eseguita con tecniche di spooling prevede una sosta intermedia dei dati diretti ai suddetti dispositivi in aree temporanee (dette aree di spooling) definite su disco. I files cosi' accodati su disco vengono poi stampati o perforati in un secondo tempo, sui dispositivi reali.
La fig. 4 schematizza questo tipo di funzionamento in due fasi. Come si vede l'operazione di I/O lanciata dalla macchina virtuale, in questo caso deve essere tradotta completamente perche' viene deviata dal Control Program su un dispositivo completamente diverso da quello a cui era originariamente destinata. Notiamo che solo la prima fase dell'operazione (quella che si esaurisce sull'area di spool) e' sincrona col programma della macchina virtuale, la seconda (da area di spool ad unita' U/R) e' differita.
La filosofia di tale organizzazione comporta la gestione di code di files secondo un criterio di FIRST-IN/FIRST-OUT. Nel momento in cui l'unita' reale termina l'operazione di I/O in corso, il primo file presente nella coda viene indirizzato su tale unita'.
Quanto detto finora e' relativo alle sole unita' di output, le cose sono perfettamente simmetriche anche per le unita' U/R di input (lettori). In questo caso ogni deck di schede che deve essere letto dalla macchina virtuale prima che possa essere disponibile per la macchina stessa deve essere scritto in area di spool.

L'Ipervisore del Control Program fornisce altri due servizi che non sono connessi con le simulazioni delle istruzioni e dei dispositivi di I/O. Esso infatti mantiene ed aggiorna i "timers" delle macchine virtuali e fornisce un'estensione all'interfaccia verso le macchine virtuali consentendo le chiamate dirette di routine di Ipervisore da programma d'utente tramite l'istruzione Diagnose.

Il gestore delle risorse e' il componente del Control Program che piu' assomiglia ai componenti corrispondenti di

supervisori tradizionali. Esso consiste a sua volta di piu' parti destinate a gestire la memoria principale, le memorie ausiliarie, l'unita' di calcolo e le operazioni di I/O reali.
La gestione della memoria centrale e' studiata per contenere, in una memoria reale limitata, soltanto le pagine di memoria virtuale piu' spesso utilizzate e per far si' che siano minimizzati i tempi di fermo delle macchine virtuali in attesa di pagine di memoria virtuale. Le richieste di memoria virtuale da parte di macchine virtuali sono soddisfatte usando la tecnica detta "a domanda di pagina" cioe' le pagine non sono lette dalle memorie ausiliarie (operazione di page-in) finche' non vengono referenziate, tuttavia il sistema mantiene un certo numero di pagine di memoria reale (page-frames) sempre libere per ricevere pagine lette da memorie ausiliarie. Un ultimo compito dei meccanismi di gestione della memoria e' il mantenimento ed aggiornamento di informazioni statistiche utilizzate dallo SCHEDULER/DISPATCHER (il componente che gestisce l'allocazione della risorsa CPU) per prevedere le richieste di memoria virtuale dei singoli utenti.
La gestione delle memorie ausiliarie affronta i problemi di allocazione dello spazio disco per operazioni di paginazione e di spooling tenendo in particolare considerazione l'efficienza dell'allocazione e la riduzione dei tempi di accesso ai dispositivi di paginazione; in particolare per i dispositivi a testine mobili si usano algoritmi che consentono di mimimizzare gli spostamenti meccanici delle testine poiche' questi influenzano pesantemente i tempi di attesa.

La gestione della risorsa CPU e' eseguita da due componenti:

a) il DISPATCHER che alloca la CPU per fette di tempo fisse e predeterminate (tecnica di time-sharing) a una "coda" di macchine virtuali basandosi su una priorita' calcolata dinamicamente che riflette la storia recente dell'utilizzo della CPU da parte di quella macchina virtuale.

b) lo SCHEDULER che gestisce la "coda" delle macchine virtuali ed usa le statistiche raccolte dalle routines di paginazione per misurare e prevedere le richieste di memoria delle singole macchine virtuali. Inoltre lo SCHEDULER suddivide le macchine virtuali in interattive (che fanno frequente I/O da terminale) e non interattive e favorisce l'accesso nella "coda" delle macchine virtuali interattive la cui esecuzione risulta pertanto favorita rispetto a quella di macchine virtuali che hanno scarsa attivita' di I/O da terminale.

Poiche' questi componenti del CP-VM (SCHEDULER/DISPATCHER) vanno in esecuzione molto freguentemente (alcune centinaia di volte per secondo) molte funzioni (addebito, previsioni di working set e di utilizzo di risorse) vengono effettuate solo quando una macchina virtuale viene estromessa (QUEUE-DROP) dalla coda quindi con frequenza molto minore.

L'ultimo componente del Control Program e' l'interfaccia verso l'utente che e' notevolmente diversa da quella dei sistemi tradizionali. Infatti essendo il VM/370 orientato prevalentemente verso l'uso di terminali, non esistono schede di Job Control Language. Secondariamente poiche' il Control Program lavora nella filosofia delle macchine virtuali il linguaggio dei comandi e' pesantemente orientato verso l'hardware del sistema piuttosto che verso l'attivita' di programmazione. Va infatti ricordato che la parte dei comandi relativi alle applicazioni risiede tutta in un altro componente del VM/370: il CMS.
Uno degli obiettivi principali seguito nell'implementazione di questo componente e' stata la semplicita' e l'intellegibilita' dei comandi (uso di blanks come delimitatori di comandi e parametri, l'uso opzionale di preposizioni come "as" o "to", etc.).
A seconda delle funzioni svolte dai comandi questo componente puo' essere suddiviso in quattro aree; tuttavia l'analisi e l'esecuzione di un comando avviene sempre attraverso le seguenti fasi:

1) L'utente mette la sua macchina virtuale in "console function" mode (che equivale in macchina reale, a fermare l'esecuzione della CPU) tramite il tasto ATTENTION (BREAK o REQUEST) del suo terminale.

2) Le routines di gestione dell'I/O da terminale leggono la riga del comando.

3) Viene selezionata la routine di esecuzione del comando; tale routine viene paginata nella memoria reale ed il comando viene eseguito.

4) La macchina virtuale riprende l'esecuzione interrotta.

Si nota che, in generale, l'esecuzione della macchina virtuale non puo' avvenire in parallelo all'esecuzione di un comando e ancora non e' ammesso che siano attivi sulla macchina virtuale piu' comandi contemporaneamente.
I comandi di Control Program sono classificabili nei seguenti sottogruppi:

1) Comandi che simulano tasti funzionali presenti sul pannello di controllo dell'elaboratore reale (IPL che simula il tasto LOAD, SYSTEM RESET, SYSTEM CLEAR che simulano i tasti omonimi).

2) Comandi che alterano lo stato dei dispositivi di I/O della macchina virtuale e controllano le funzioni di spooling.

3) Comandi che consentono di modificare la configurazione della macchina virtuale definendo, attaccando, staccando dispositivi virtuali, ridefinendo la memoria virtuale, etc.

4) Comandi per la messa a punto di programmi quali i punti di arresto e le facilities di traccia.

IL DIRECTORY DELLE MACCHINE VIRTUALI

Per gestire le macchine virtuali e simularne i dispositivi il Control Program deve fare riferimento ad un catalogo dove sono descritte tutte le macchine virtuali.
Tale catalogo, chiamato DIRECTORY, risiede su disco ed e' aggiornato dall'amministratore di sistema che, usando programmi di utilita', traduce le definizioni e le caratteristiche delle macchine virtuali, in records di DIRECTORY.

Il CP-VM, quando la situazione lo richiede, accede alle informazioni di directory:

1) per definire una nuova macchina virtuale quando questa si collega da un terminale;

2) per modificare la configurazione di una macchina virtuale gia' esistente;

3) per verificare l'esistenza di una macchina virtuale.

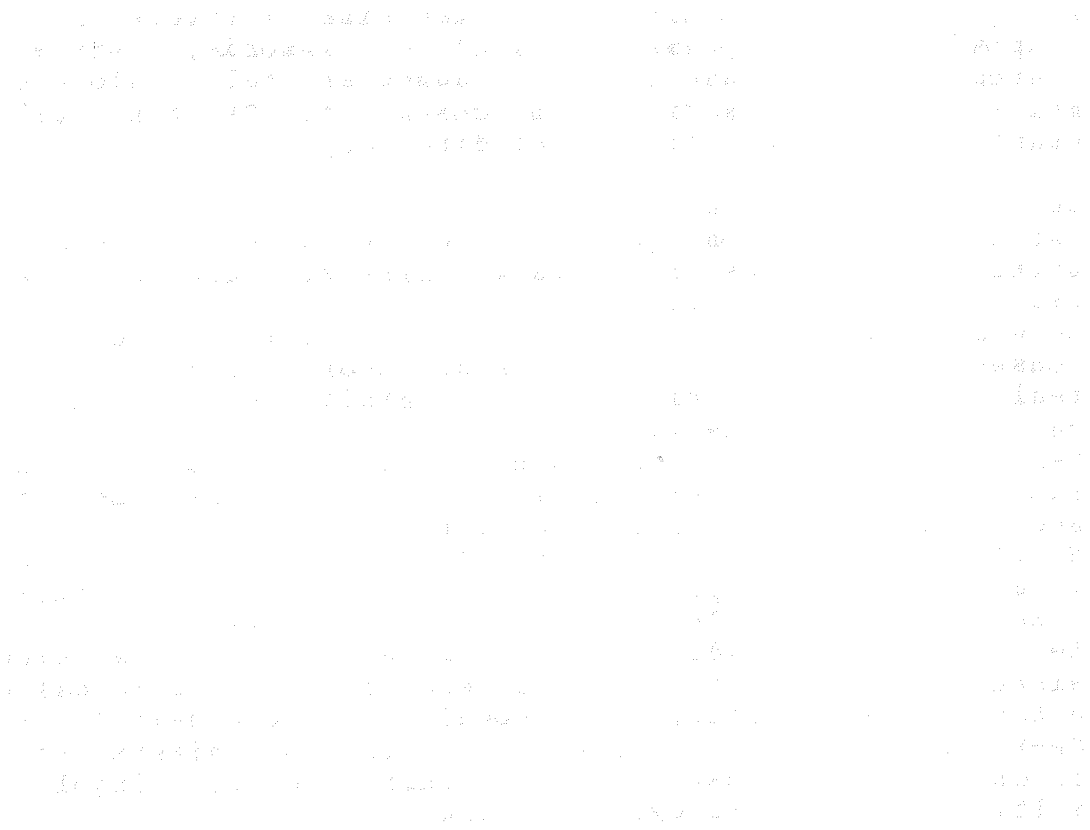
Ogni entry nel directory e' associata ad un utente e contiene il nome della macchina virtuale (USERID), la parola chiave (PASSWORD) ed altre informazioni che descrivono la configurazione della macchina stessa e che sono richieste dall'utente per poter svolgere il proprio lavoro.
Al momento del collegamento da terminale (LOGIN) riconosciuta la USERID e la PASSWORD corrette, il sistema crea una macchina virtuale basandosi sulle informazioni del directory.
Le principali componenti della configurazione di una macchina virtuale sono:

1) Console virtuale
   Simulata su terminali. Usando la tastiera ed i comandi di CP, un utente puo' eseguire la maggior parte delle funzioni che un operatore puo' svolgere sulla macchina reale.

2) Memoria virtuale
   Ogni macchina virtuale ha definito nel directory la quantita' di memoria virtuale che va da un minimo di 8K bytes ad un massimo di 16M bytes. Nel directory sono riportate due dimensioni di memoria che si riferiscono rispettivamente alla quantita' di memoria fornita normalmente ed alla quantita' massima di memoria che si puo' utilizzare.

Al momento del LOGIN, una macchina virtuale ha a disposizione la prima quantita' di memoria, tuttavia l'utente e' in grado di poter aumentare tale valore di memoria virtuale mediante un comando di CP, fino alla quantita' massima definita nel directory.

3) Unita' di I/O virtuali
Nel directory vengono generalmente definite tutte quelle unita' di I/O necessarie alla macchina virtuale per poter svolgere il proprio lavoro.
Generalmente ogni utente ha uno spazio disco dove conservare i propri files (un minidisco). Quindi un disco reale puo' trovarsi diviso in piu' minidischi ciascuno ad una o piu' maccchine virtuali.
Per ogni minidisco definito nella configurazione di una macchina virtuale e' possibile stabilire il modo di accesso mediante un'opzione di directory.
Se il minidisco appartiene ad un'unica macchina virtuale, il modo di accesso puo' essere di due tipi: sola lettura (READ/ONLY) oppure lettura-scrittura (READ/WRITE).
Se invece il minidisco e'in comune fra piu' macchine virtuali, il modo di accesso dipende dall'uso fatto dalle altre macchine virtuali che condividono tale unita'. Il CP-VM permette di specificare nel directory diversi tipi di opzioni che consentono di definire accesso singolo o multiplo, in lettura e/o scrittura.



PARTE TERZA:

DESCRIZIONE DEL COMPONENTE CMS

Una macchina virtuale e' la simulazione di un sistema S/370, per cui, pur con le limitazioni citate nel paragrafo "Restrizioni al funzionamento in macchina virtuale", un qualunque sistema operativo di quelli elencati in figura 1 puo' esservi caricato. Ad ogni macchina virtuale che non lavora in modo disconnesso, e' associato un terminale, che simula la console dell'operatore del sistema/370 virtuale. In questo modo il CP realizza un sistema di time-sharing fra i terminali conversazionali che simulano la console dell'operatore. Per realizzare un servizio di time-sharing in ambiente VM, la soluzione piu' semplice e' quindi quella di assegnare ad ogni utente una macchina virtuale, in modo che ogni utente sia l'operatore della sua macchina virtuale. I sistemi operativi elencati in figura 1, pero', non si prestano ad essere utilizzati da un unico utente che ne sia anche l'operatore sia perche' un generico utente non ha nessun motivo di imparare il linguaggio dei comandi dell'operatore, sia perche' in tali sistemi operativi l'operatore il piu' delle volte non ha la possibilita' di passare programmi, sia, dal punto di vista delle prestazioni, perche' molte funzioni svolte dai sistemi operativi tradizionali sono gia' realizzate dal CP.
Il CMS e' stato progettato per essere utilizzato sotto macchina virtuale da un unico utente; nel paragrafo che segue vengono esaminate le ragioni che rendono il CMS preferibile in un ambiente di macchine virtuali sia dal punto di vista delle prestazioni sia dal punto di vista della semplicita' di uso.

VANTAGGI DEL CMS IN UN AMBIENTE A MACCHINE VIRTUALI

Il CMS e' stato progettato come sistema destinato a girare su macchine virtuali in cui l'unico utente sia lo stesso operatore; questo fatto spiega numerose scelte che differenziano il CMS dagli altri sistemi operativi e che sono elencate nel presente paragrafo.

Time-sharing:
Il servizio di time-sharing viene svolto dal CP fra i terminali che sono consoles di macchine virtuali. Il CMS deve pertanto gestire un solo terminale, precisamente la console del suo operatore, per cui non e' in grado di fornire un servizio di time-sharing. Ovviamente, e' possibile attaccare ad una macchina virtuale CMS piu' di un terminale, ma spetta in tal caso al programmatore scrivere il software necessario a gestire i terminali supplementari. In questo modo si possono scrivere applicazioni di tipo Transaction Processing sotto CMS, pur non potendosi utilizzare a tal fine i packages standard (CICS, TCAM, BTAM, VTAM etc.) distribuiti dall'IBM.

Job-Entry:
Essendo il CMS un sistema conversazionale, i programmi vengono mandati in esecuzione per mezzo di comandi introdotti dal terminale e non si puo' parlare di jobs. Esiste pero' una varieta' del CMS (il CMS BATCH), che accetta i comandi, sempre in modo sequenziale, dal lettore di schede della macchina virtuale anziche' dal terminale. In questo caso esiste un comando, attivabile per mezzo della scheda "/JOB", che serve a riportare il CMS nelle condizioni iniziali e a far addebitare le elaborazioni che seguono ad un nuovo codice. In questo ambiente viene dato il nome job alle elaborazioni eseguite fra due schede "/JOB" successive. Naturalmente, anche nel CMS BATCH, non e' possibile parlare di software preposto all'ingresso dei jobs, in quanto tutto si riduce alla lettura di una nuova scheda quando termina l'esecuzione di un comando. Similmente, non c'e' nulla in CMS preposto al Remote Job Entry; infatti, i terminali batch remoti di un sistema VM/370 sono normalmente attaccati alla macchina virtuale su cui gira il sistema RSCS, e questo provvede ad inviare le schede che riceve in input dai lettori remoti sui lettori virtuali delle macchine virtuali opportune. Il CMS non puo' quindi distinguere le schede provenienti da un lettore locale da quelle provenienti da un remoto. Discorsi simili possono essere ripetuti per le uscite su perforatore o su stampante.

Multiprogrammazione:
Essendo il CMS un sistema per un solo utente, non esistono possibilita' di multiprogrammazione. Non e' quindi possibile scrivere programmi composti di piu' processi paralleli, a meno di non includere nel programma stesso la componente destinata alla supervisione dei processi. Questo puo' essere considerato uno svantaggio in quanto limita la possibilita' di mettere a punto in CMS programmi destinati ad altri sistemi, ma bisogna altresi' riconoscere che la suddivisione di un programma in piu' processi paralleli non darebbe in CMS nessun miglioramento delle prestazioni, in quanto la distribuzione del tempo di CPU viene fatta dal CP a livello di macchine virtuali.

Paging e Spooling:
Tutte le operazioni di paging e di spooling sono svolte dal CP. Per quanto concerne la paginazione, il CMS lavora con indirizzi reali, immaginando di trovarsi su un 370 reale con dimensioni di memoria uguale a quella della macchina virtuale.
Similmente, il CMS non gestisce aree di spool, ma permette a tutti i programmi di leggere schede direttamente dal lettore della macchina virtuale e di scrivere sul perforatore e la stampante virtuali.
Una conseguenza di questo fatto si nota nel caso di programmi con piu' files di uscita destinati alla stampatrice: mentre nei sistemi operativi con spool (come l'OS/HASP) ogni file viene stampato separatamente, in CMS le righe di stampa compaiono in uscita esattamente nello stesso ordine in cui sono state scritte dal programma, indipendentemente dal file a cui erano destinate.

Entrata/Uscita via diagnose:
Nei sistemi S/370, la CPU per mezzo dell'istruzione SIO, fa in modo che il canale inizi un'operazione di I/O.
A sua volta, il canale segnala alla CPU la fine di un'operazione per mezzo di un'interruzione di I/O. Generalmente i sistemi operativi utilizzano il tempo fra la SIO e l'interruzione di I/O passando il controllo ad altri processi. L'interruzione di I/O viene trattata verificando la corretta esecuzione dell'operazione e segnalando che il processo che aveva dato inizio all'operazione puo' nuovamente ricevere il controllo. In un ambiente a macchine virtuali, questo meccanismo comporta la duplicazione del trattamento delle interruzioni di I/O, in quanto si deve avere sia il trattamento dell'interruzione reale, eseguito dal CP, sia il trattamento dell'interruzione virtuale, eseguito dal sistema operativo della macchina virtuale.
Nel CP, tuttavia, e' previsto che una macchina virtuale possa dare inizio ad un'operazione di I/O non solo per mezzo

di una SIO, ma anche per mezzo dell'istruzione "diagnose". In questo caso il CP restituisce il controllo alla macchina virtuale solo al termine dell'operazione di I/O, per cui e' possibile fare a meno dell'interruzione virtuale di fine operazione. Per la macchina virtuale l'istruzione "diagnose" dura quanto l'operazione di I/O e, quando viene eseguita l'operazione successiva, i dati sono stati sicuramente trasferiti. In un sistema operativo che non ha altri processi cui cedere il controllo quando parte un'operazione, il metodo della "diagnose" permette di sopprimere le routines per il trattamento delle interruzioni, ottenendo una semplificazione dell'architettura del sistema e, almeno in parte, un miglioramento delle prestazioni.
Il CMS esegue tutte le operazioni di entrata-uscita destinate a nastri e dischi per mezzo della istruzione "diagnose". Nel complesso, questo fatto non comporta conseguenze dal punto di vista dell'utente. Le uniche occasioni in cui la presenza della "diagnose" puo' essere osservata sono le operazioni di I/O sui nastri, che possono durare anche molti minuti. In tutto questo tempo il CMS esegue una sola istruzione, per cui non e' interrompibile ed inviando delle "attention" da terminale si riesce solo ad entrare in CP. Un fatto simile accade anche quando un'unita' nastro non e' "ready", nel qual caso il CP restituisce il controllo alla macchina virtuale solo quando l'unita' e' stata rimessa "ready" e l'operazione sospesa ha potuto essere condotta a termine.

Segmenti shared:
Il CP offre la possibilita' a piu' macchine virtuali di avere segmenti di memoria (cioe' pezzi di 64K contigui aventi inizio da indirizzi multipli di 64K) in comune e accessibili in sola lettura.
Se una macchina virtuale tenta di modificare il contenuto di uno di questi segmenti, la modifica viene fatta su una copia del segmento e la macchina virtuale responsabile lavora da questo momento in poi sulla copia senza danneggiare il funzionamento delle altre macchine virtuali. I segmenti shared sono difficilmente utilizzabili nei sistemi operativi tradizionali, perche' raramente questi hanno interi segmenti programmati in modo rientrante. Il CMS e' stato scritto tenendo conto di queste possibilita'. Tutte le macchine che fanno IPL per mezzo del comando IPL CMS hanno il secondo segmento (cioe' quello che inizia all'indirizzo esadecimale 10000) in comune. Questo segmento contiene quasi tutto il nucleo del CMS. Le macchine che fanno IPL 190 lavorano invece con una copia privata di tutto il CMS. All'indirizzo esadecimale 100000 si trova il segmento chiamato CMSSEG, che contiene l'editore, l'interprete delle procedure EXEC ed il

simulatore delle macro di OS. Il segmento CMSSEG puo' essere utilizzato solo dalle macchine virtuali con memoria non superiore ad 1M, altrimenti il suo contenuto potrebbe essere alterato nel corso delle elaborazioni. Nelle macchine virtuali con piu' di 1M di memoria, i programmi che compongono il CMSSEG vengono caricati automaticamente con il comando LOAD all'indirizzo piu' alto possibile in fase di inizializzazione. Altri segmenti shared disponibili in CMS sono il DOS, il VSAM, l'AMS ed il VSAPL.

Moduli non rilocabili:
I compilatori producono generalmente il programma oggetto nella forma di files contenenti codice rilocabile. Questi files contengono il programma scritto in linguaggio macchina e varie informazioni di controllo, che servono a risolvere gli agganci con altri programmi compilati separatamente ed a fare in modo che il programma possa essere caricato a qualunque indirizzo. L'interpretazione di queste informazioni di controllo rende il caricamento dei programmi piuttosto lento. In CMS, se un programma non ha bisogno di risolvere agganci con altri programmi non c'e' nessuna necessita' di poter caricare il programma a qualunque indirizzo: questa esigenza e' infatti fondamentale solo nei sistemi operativi in cui la memoria viene spartita fra piu' utenti. Per queste ragioni, il CMS permette anche l'uso di files contenenti codice non rilocabile, da caricare sempre allo stesso indirizzo di memoria.

Configurazione del sistema:
I sistemi operativi hanno bisogno di conoscere la configurazione delle apparecchiature periferiche del sistema in cui operano. Per questa ragione, durante la generazione di un sistema operativo, si deve fornire come dati la lista degli indirizzi delle apparecchiature e dei loro tipi. La generazione del CMS, pero', non viene fatta dai singoli possessori delle macchine virtuali (i quali, fra l'altro, non hanno nessun motivo di imparare a generare i sistemi operativi), ma tutti gli utenti di CMS di un dato centro lavorano con la stessa copia di CMS, la cui generazione e' compito di chi ne cura la manutenzione. Ciononostante, il CMS permette una discreta flessibilita' nella configurazione della macchina virtuale, grazie ai seguenti accorgimenti:

- L'indirizzo della console della macchina virtuale viene richiesto al CP in fase di inizializzazione per mezzo di un'apposita "diagnose".

- La stessa istruzione "diagnose" viene utilizzata per conoscere il tipo di un device di cui si conosce l'indirizzo e verificarne l'esistenza tutte le volte che

e' necessario.

- I minidischi possono avere qualunque indirizzo. Nei comandi di CMS, pero', i minidischi vengono indicati, anziche' per mezzo del loro indirizzo, per mezzo di una lettera, detta la "mode letter". Per mezzo del comando di CMS ACCESS, l'utente associa una "mode letter" al minidisco con un dato indirizzo; in questa occasione il CMS lancia la "diagnose" per conoscere il tipo del minidisco.

- Alcune appareccciature virtuali hanno indirizzi fissi. Queste sono:

  1) il lettore di schede (00C);
  2) il perforatore di schede (00D);
  3) la stampatrice (00E);
  4) le unita' a nastri (181, 182, 183 e 184)

  Le unita' a nastri vengono attaccate ad una macchina virtuale solo al momento dell'uso, per cui l'obbligo di attaccarle con uno dei quattro indirizzi elencati sopra non costituisce una grave limitazione. Per quel che riguarda le apparecchiature di I/O su schede o su stampante, la limitazione, piu' che nella fissita' degli indirizzi, va vista nell'incapacita' del CMS di gestire piu' di un perforatore, un lettore o una stampatrice.
  L'utente di CMS puo' utilizzare anche apparecchiature non previste dal sistema standard, purche' egli stesso fornisca i programmi che gestiscono l'I/O su tali apparecchiature.

Gestione dello spazio-disco:
Generalmente, nei sistemi operativi IBM, lo spazio disco e' disponibile a tutti gli utenti. Quando questi vogliono memorizzare un insieme di dati su disco, devono specificare il nome dell'insieme di dati e lo spazio necessario a contenerli (allocazione del data-set). In questo modo il sistema operativo dedica fino al momento della cancellazione un certo gruppo di records all'insieme di dati con quel nome.
In un sistema conversazionale e' normale utilizzare insiemi di dati su disco con frequenza molto maggiore che nei sistemi batch. La necessita' di allocare i data-set, quindi, appesantisce molto l'uso del sistema. Inoltre, l'utente, che alloca un data set, e' sempre portato a dichiarare molto piu' spazio di quello di cui ha effettivamente bisogno, cosa che deve essere assolutamente evitata in un sistema in cui generalmente si lavora con minidischi di pochi cilindri. Per questi motivi, l'allocazione dello spazio-disco non e'

necesaria in CMS, ma i dati vengono memorizzati dal sistema operativo su disco secondo un metodo estremamente particolare: i files, qualunque sia il loro formato, vengono memorizzati su records fisici di 800 caratteri; sia in memoria sia su disco, il sistema operativo mantiene una mappa dell'occupazione dei blocchi di 800 bytes, dalla quale il CMS puo' sapere su quali records possono essere scritti i nuovi dati; un insieme di dati, quindi, non viene memorizzato su records fisicamente adiacenti ma puo' trovarsi sparso per tutto il minidisco; per ritrovare un insieme di dati, il CMS mantiene una struttura di puntatori ad albero, anch'essa memorizzata su disco. Il metodo di gestione dello spazio-disco del CMS permette di sfruttare in modo ottimale i minidischi riducendo al minimo le indicazioni che l'utente deve fornire.
L'unica conseguenza negativa di questo metodo e' l'impossibilita' di usare lo stesso minidisco in lettura-scrittura da due macchine virtuali diverse.

ORGANIZZAZIONE DEI DATI SU DISCO

Prima di proseguire con la spiegazione del funzionamento del CMS, e' bene descrivere il modo in cui i dati vengono registrati sui minidischi. In un determinato istante l'utente puo' avere accesso al contenuto di 10 minidischi al massimo. Ad ogni minidisco disponibile e' associata una "mode-letter".
Le mode-letters ammesse sono A,B,C,D,E,F,G,S,Y e Z. Una mode-letter viene associata ad un minidisco per mezzo del comando ACCESS. Gli insiemi di dati sui minidischi vengono sempre chiamati, usando la terminologia del CMS, files. I programmi che girano sotto CMS, per manipolare un file, ne devono fornire il nome, che e' costituito da una terna di identificatori, detti filename, filetype e filemode. Il filename ed il filetype vengono scelti dall'utente che crea il file e sono successioni di otto caratteri al massimo. Il filemode e' una coppia di caratteri, il primo dei quali e' la "mode-letter" del minidisco su cui risiede il file; il secondo carattere e' una cifra compresa fra 0 e 6, il cui significato e' legato al modo in cui e' possibile accedere al file. Per il normale accesso in lettura-scrittura la cifra del filemode e' 1, e questa e' anche la cifra che viene automaticamente assegnata dal sistema se l'utente non la specifica al momento della creazione del file. Nei cataloghi dei minidischi sono registrati il filename, il filetype e la cifra del filemode; manca, ovviamente, la lettera del filemode, perche' questa viene associata ai files di un minidisco solo al momento del comando ACCESS. Volendo verificare l'esistenza di un file e' possibile specificare un "*" al posto del filemode. In questo caso il file viene ricercato su tutti i minidischi disponibili secondo l'ordine alfabetico delle mode-letters. Il programma che voleva verificare l'esistenza del file puo', in questo modo, conoscere la mode-letter del primo minidisco in ordine alfabetico contenente il file con il nome e tipo richiesto. Questo fatto e' fondamentale perche', come si vedra', da' la possibilita' agli utenti di definire copie personali di comandi di CMS. L'ordine di ricerca dei files sui minidischi viene normalmente detto ordine standard di ricerca (standard order of search).
Come si e' detto, tutti i files vengono registrati dal CMS su blocchi fisici di 800 caratteri. I formati logici dei files di CMS sono due: fisso (F) e varaibile (V). Il formato variabile si differenzia dal fisso per il fatto che tutti i records logici sono preceduti da 2 bytes contenenti la lunghezza del record. Le funzioni del CMS per la lettura e scrittura su file permettono:

a- la lettura di n records a partire dall'n-esimo record di un file in formato fisso;
b- la lettura di 1 record a partire dall'n-esimo record di un file in formato variabile;
c- la scrittura di n records a partire dall'n-esimo record di un file in formato fisso (con la possibilita' di lasciare dei "vuoti" all'interno del file);
d- la scrittura di 1 record in fondo ad un file in formato variabile.

Altre funzioni del CMS permettono la modifica del catalogo di un minidisco senza alterare il contenuto dei files. Esempi di queste funzioni sono la cancellazione di un file (ERASE) e l'alterazione del suo nome (RENAME).
Si e' detto che la scelta del filetype e' libera. In realta', in molti comandi di CMS, l'utente deve specificare solo il filename dei files in ingresso, mentre il filetype viene deciso dal comando stesso. La scelta del filetype, quindi, e' motivata il piu' delle volte dall'uso che si intende fare del file. Alcuni filetypes importanti sono:

TEXT - e' il filetype dei files prodotti dai compilatori, contenenti codice oggetto rilocabile. Il loader del CMS (comando LOAD) limita la sua ricerca ai files di tipo TEXT e TXTLIB (di questi ultimi si parla piu' avanti).
MODULE- e' il filetype dei files contenenti codice oggetto non rilocabile. I files con questo tipo vengono ricercati dal comando LOADMOD.
EXEC - e' il filetype dei files contenenti procedure di comandi di CMS. I files con questo tipo vengono ricercati dal comando EXEC.

I comandi di CMS MACLIB e TXTLIB permettono la creazione e la manipolazione di files partizionati, ossia files che contengono al loro interno un catalogo di membri. Questi due comandi trattano solo files in formato scheda. Con il comando TXTLIB si possono creare o manipolare files di tipo TXTLIB; questi files vengono utilizzati dal loader del CMS per la ricerca dei sottoprogrammi di libreria. Il comando MACLIB manipola files di tipo MACLIB; questi files permettono di simulare il metodo di accesso BPAM dell'OS.

FUNZIONAMENTO DEL CMS

In assenza di attivita', il CMS attende un comando da terminale. Il comando viene dato sotto forma di una serie di una o piu' parole lunghe al massimo 8 caratteri. La prima parola viene detta "nome del comando" e viene utilizzata dal CMS per determinare a quale programma deve essere passato il controllo. La ricerca del comando avviene nel seguente modo: (*)

- Se su uno dei minidischi disponibili esiste un file con nome uguale a quello del comando e tipo EXEC, la parola EXEC viene posta davanti al nome del comando, cosi' che EXEC diventa il nuovo nome.
- Con il registro 1 che punta al comando, viene lanciata una SVC 202; questo metodo puo' essere utilizzato in qualunque programma di CMS, per fare eseguire al sistema qualche comando.
- Il programma che tratta la SVC 202 esegue dapprima una ricerca in una tabella residente nel nucleo, nella quale e' contenuto l'elenco dei comandi di CMS residenti permanentemente in memoria e dei loro indirizzi; se il comando viene trovato in questa tabella, gli viene passato il controllo; altrimenti viene guardata una locazione in memoria, che contiene il nome dell'ultimo file di tipo MODULE che e' stato caricato in "area transiente" (cioe', alla locazione esadecimale X'E000'); se questo nome e' il nome del comando, il controllo viene passato alla locazione X'E000'; altrimenti, viene fatta una ricerca sui minidischi disponibili, per vedere se esiste un file di tipo MODULE con nome uguale al nome del comando; se il file esiste, viene caricato in memoria dalla routine LOADMOD, dopo di che gli viene passato il controllo.

Se nessuna delle tre ricerche ha successo, il programma delle SVC 202 considera il comando invalido e si prepara a darne segnalazione al programma chiamante caricando il numero -3 nel registro 15. Anche nel caso in cui una delle ricerche abbia avuto successo, il programma chiamato segnala alla SVC 202 il successo o l'insuccesso del comando caricando rispettivamente 0 o un numero positivo nel registro 15. In generale, quindi, al momento del ritorno dal programma della SVC 202 al programma chiamante il contenuto

---

(*) per semplicita', si e' omesso il trattamento delle abbreviazioni dei comandi.

del registro 15 ha il seguente significato.

0 - comando eseguito correttamente
>0 - comando eseguito con errori
<0 - comando non trovato

- Al ritorno dalla SVC 202, il CMS controlla il contenuto del registro 15. Se questo e' positivo o nullo, viene eseguito il passo che segue. Altrimenti, il CMS passa il comando al CP per mezzo di una "diagnose"; se il comando non e' conosciuto nemmeno al CP, viene caricato il numero -1 nel registro 15; altrimenti, viene lasciato nel registro 15 il valore impostato dal CP, che e' sempre un numero positivo o nullo.
- A questo punto, il CMS deve segnalare all'utente la fine del comando e la disponibilita' ad accettarne un altro. La segnalazione avviene nel seguente modo:

  a) Se il contenuto del registro 15 e' negativo, viene stampato il messaggio:

     UNKNOWN CP/CMS COMMAND

  b) Se il contenuto del registro 15 e' nullo, viene stampato il "ready message" nella forma:

     R; T=a.bb/c.dd hh:mm:ss,

     dove a.bb indica la durata del comando in secondi e centesimi di secondo di CPU virtuale, c.dd la durata in secondi e centesimi di secondo di CPU totale e hh:mm:ss l'ora di fine del comando.

  c) Se il contenuto del registro 15 e' positivo, viene stampato il "ready message" nella forma:

     R(nnnnn); T=a.bb/c.dd hh:mm:ss,

     dove nnnnn e' il contenuto del registro 15 al termine del comando.

Talvolta, errori particolarmente gravi impediscono al CMS di portare a termine il comando nel modo usuale; in questo caso, il CMS dichiara la sua disponibilita' a ricevere un nuovo comando stampando a terminale il messaggio:

CMS

A questo tipo di terminazione si da' il nome di ABEND (da ABnormal END). L'ABEND puo' essere provocato volutamente

dall'utente per mezzo del comando immediato HX. Dopo un ABEND, il CMS suppone che buona parte dei dati contenuti in memoria siano invalidi, per cui pulisce varie aree e considera nuovamente disponibili alcune zone di memoria allocate dinamicamente. A questa operazione si da' il nome di Abend Recovery. L'Abend Recovery viene eseguita quando l'utente introduce il primo comando da terminale dopo il messaggio CMS. Se pero' a questo punto viene dato il comando DEBUG, L'Abend Recovery non viene eseguita subito, per dare modo all'utente di esaminare il contenuto della memoria al momento in cui si e' verificato l'errore. Ritornando dal DEBUG al CMS, il "ready message" indica che non e' stata eseguita l'Abend Recovery, mentre il messaggio CMS ne segnala l'avvenuta esecuzione.
In base alle alterazioni che causano nel contenuto della memoria della macchina virtuale, i comandi di CMS possono essere suddivisi in tre categorie:

a - comandi che quando terminano non lasciano modifiche nel contenuto della memoria. La maggior parte dei comandi di CMS appartiene a questa categoria.
b - Comandi che lasciano in memoria modifiche che vengono annullate dall'Abend Recovery. Appartengono a questa categoria i comandi DLBL, FILEDEF, LABELDEF, SVCTRACE.
c - Comandi che lasciano in memoria modifiche che vengono annullate dal SYSTEM RESET (e, di conseguenza, dall'IPL). Appartengono a questa categoria i comandi: ACCESS, ASSGN, FETCH, GLOBAL, INCLUDE, LOAD, LOADMOD, OPTION, RELEASE, SET, SYNONYM.

Quando il CMS si prepara ad attendere il primo comando dopo l'IPL, l'unico minidisco disponibile (ossia quello il cui catalogo e' noto al sistema operativo) e' il disco-sistema, che normalmente ha l'indirizzo 190. I files che si trovano su questo minidisco hanno filemode S e sono disponibili in sola lettura. Tutte le macchine virtuali che lavorano in CMS usano generalmente lo stesso minidisco 190, sul quale vengono conservati i comandi e le librerie di sistema residenti su disco. L'associazione minidisco-modeletter che si ottiene per mezzo del comando ACCESS puo' essere annullata per tutte le modeletters diverse da S per mezzo del comando RELEASE o per mezzo di un nuovo comando ACCESS. L'associazione 190-S, invece, viene fatta dal CMS in fase di inizializzazione e non puo' essere mai annullata; questo fatto assicura la continua disponibilita' dei comandi e delle librerie di sistema residenti su disco.
Se il primo comando dato dopo l'IPL e' diverso da ACCESS, il CMS, prima di eseguirlo, emette automaticamente i seguenti comandi:

ACCESS 19E Y/S * * Y2
ACCESS 191 A
ACCESS 192 D (o FORMAT 192 D se il minidisco 192 e' un "temporary disk" non ancora in formato CMS).
Naturalmente questi comandi vengono eseguiti solo se nella configurazione della macchina virtuale esistono i minidischi con gli indirizzi suddetti. Se il primo comando dato dopo l'IPL e' ACCESS, questo viene eseguito al posto del comando ACCESS 191 A; se tale comando fa riferimento all'indirizzo 192 o alla mode letter D, il comando ACCESS 192 D non viene eseguito. In questa fase, dunque, il CMS acquisisce il catalogo di tre minidischi al massimo. Il primo dei tre minidischi viene detto estensione del disco-sistema; come il disco-sistema, questo e' un minidisco disponibile a tutte le macchine virtuali in sola lettura. Generalmente l'estensione del disco-sistema viene utilizzata per contenere i programmi di interesse comune che non vengono distribuiti col CMS (ad esempio, i compilatori). Il secondo minidisco viene detto disco primario. Si prevede che l'utente utilizzi questo minidisco per conservarvi i suoi files personali. Il terzo minidisco viene detto temporaneo; la sua presenza e' particolarmente vantaggiosa se il minidisco di indirizzo 192 e' un "temporary disk" anche per il CP. Per "temporary disk" si intende un minidisco che viene definito ed assegnato dal CP alla macchina virtuale solo quando viene richiesto da quest'ultima; il minidisco, in altre parole, non occupa una posizione fissa su un disco reale, ma viene definito quando serve in una zona destinata dal CP ai "temporary disk". Quando un "temporary disk" viene staccato dalla configurazione di una macchina virtuale per mezzo dei comandi DETACH o LOGOFF, i cilindri che lo componevano ritornano disponibili per altre macchine virtuali, per cui il loro contenuto non puo' essere recuperato. Il disco temporaneo in CMS e' previsto essenzialmente per contenere files di lavoro la cui conservazione non e' necessaria.
Terminata questa fase, se il comando ACCESS relativo al disco primario e' stato eseguito automaticamente o e' stato dato dall'utente senza l'opzione NOPROF, il CMS guarda se sul minidisco primario esiste il file PROFILE EXEC e, se esiste, emette il comandO:

```
EXEC PROFILE
```

A questo punto, termina l'inizializzazione del CMS e compare il ready message. Se il comando dato subito dopo l'IPL non era ACCESS, viene eseguito in questo momento, causando, al suo termine, la comparsa di un secondo ready message.

GESTIONE DELLA MEMORIA DEL CMS

Il CMS utilizza la memoria della macchina virtuale suddividendola nelle seguenti porzioni:

Dalla locazione X'000000' alla X'003000': Modulo DMSNUC. Questo modulo appartiene alla parte del CMS permanentemente residente in memoria, e contiene tutti gli insiemi di dati che, essendo diversi dall'una all'altra macchina virtuale, non possono essere inclusi nella parte "shared" del nucleo.

Dalla locazione X'003000' alla X'00E000': Spazio riservato all'allocazione dinamica della memoria. Il modo in cui avvengono le allocazioni viene spiegato in dettaglio piu' avanti.

Dalla locazione X'00E000' alla X'010000': Area transiente. Gran parte di files di tipo MODULE (programmi oggetto non rilocabili) corrispondenti a comandi di CMS vengono caricati in quest'area. Il caricamento di questi moduli, di conseguenza, non ricopre cio' che si trova nell'area riservata ai programmi di utente, rendendo possibile la chiamata da parte di questi ultimi di buona parte dei comandi di CMS. Il caricamento di un programma oggetto rilocabile in area transiente e' possibile usando i comandi LOAD o INCLUDE con l'opzione ORIGIN TRANS.

Dalla locazione X'010000' alla X'020000': Nucleo rientrante. Contiene tutti i moduli di CMS permanentemente residenti in memoria, ad eccezione di DMSNUC. Tutti questi programmi sono scritti in modo rientrante, cioe', in modo tale che la loro esecuzione non ne modifica il contenuto. In altre parole, il CMS ha bisogno dei programmi contenuti in questo segmento in sola lettura, per cui il segmento puo' essere condiviso fra piu' macchine virtuali, cosa che effettivamente avviene se il CMS viene caricato inizialmente con il comando IPL CMS.

Dalla locazione X'020000' alle tabelle del Loader: Area per i programmi di utente. Questo spazio e' disponibile sia per l'allocazione dinamica della memoria sia per il caricamento di programmi per mezzo dei comandi LOAD e INCLUDE. Il caricamento avviene a partire dalla locazione X'020000' nel caso del comando LOAD o a partire dalla prima doppia voce disponibile dopo l'ultimo programma caricato nel caso del comando INCLUDE. Con entrambi i comandi e' possibile modificare l'indirizzo di caricamento dei programmi per mezzo dell'opzione ORIGIN. In particolare, con l'opzione ORIGIN TRANS, il caricamento avviene alla locazione

X'00E000'. I files di tipo MODULE (programmi oggetto non rilocabili) vengono generati per mezzo del comando GENMOD dopo che il programma e' stato caricato in memoria con il comando LOAD. Il MODULE cosi' generato, potra' in seguito essere caricato solo all'indirizzo che occupava al momento della generazione. E' chiaro, quindi, che esistono files di tipo MODULE destinati ad essere caricati nell'area per i programmi di utente. In particolare, anche alcuni comandi di sistema del CMS vengono caricati in quest'area: generalmente si tratta dei comandi che o occupano piu' di due pagine di memoria, e quindi non possono essere contenuti nell'area transiente, o chiamano altri comandi di CMS caricabili in area transiente. La chiamata di un modulo destinato all'area per programmi di utente da un programma situato in quest'area provoca infatti il ricoprimento del programma chiamante. Al sottoinsieme di comandi chiamabili da un programma situato nell'area per programmi di utente viene dato il nome di CMS SUBSET.

Dall'inizio delle tabelle del Loader alla fine della memoria virtuale: Tabelle del Loader. Questo spazio e' utilizzato dal Loader per costruire le tabelle relative ai simboli esterni incontrati nel corso del caricamento. Ogni tabella occupa 20 bytes e contiene il nome del simbolo esterno, il suo indirizzo di caricamento ed altre informazioni. Essendo quest'area riservata esclusivamente alle tabelle del Loader, il suo contenuto si conserva fra un comando e l'altro e in occasione degli ABEND. Un comando di CMS puo' quindi servirsi di informazioni inserite nelle tabelle del Loader nel corso di un precedente comando. In particolare, la differenza fra i comandi LOAD e INCLUDE sta nel fatto che il comando LOAD inizia distruggendo il contenuto delle tabelle, mentre il comando INCLUDE lo conserva e se ne serve nel corso del caricamento. Quando si genera un file di tipo MODULE, usando l'opzione MAP nel comando GENMOD si fa in modo che il contenuto delle tabelle del Loader sia copiato nell'ultimo record logico del file. Quando un MODULE generato in questo modo viene caricato in memoria, il contenuto del suo record logico viene copiato nelle tabelle del Loader, cosi' che un eventuale comando INCLUDE dato successivamente puo' risolvere agganci con simboli esterni definiti nel MODULE. Per i MODULE generati in area transiente non e' possibile usare l'opzione MAP.

Si ricorda infine che il CMS puo' utilizzare inoltre segmenti di memoria in comune con altre macchine virtuali situati ad indirizzi maggiori della memoria virtuale della macchina. Le allocazioni dinamiche di memoria si possono ottenere in CMS per mezzo di due tipi di richieste. Un tipo di allocazione si ottiene con l'uso delle macro di OS

GETMAIN e FREEMAIN. Le aree richieste da queste macro vengono allocate alla prima locazione disponibile dopo l'ultimo programma caricato, procedendo per indirizzi crescenti. Quando termina un comando di CMS, tutte le aree allocate con la macro GETMAIN sono considerate nuovamente libere. L'altro tipo di allocazione si ottiene con l'uso delle macro di CMS DMSFREE e DMSFRET. Le aree richieste da queste macro vengono allocate procedendo per indirizzi decrescenti utilizzando finche' e' possibile lo spazio compreso fra l'area transiente e DMSNUC e in seguito l'area per i programmi di utente, sempre partendo dall'indirizzo piu' alto. Le aree allocate con la macro DMSFREE sono di due tipi: quelle di tipo USER, che vengono considerate nuovamente libere se si verifica un ABEND, e quelle di tipo NUCLEUS, che vengono conservate anche dopo gli ABEND. Le aree di tipo NUCLEUS sono pero' protette in scrittura, per cui possono essere utilizzate solo rinunciando alla protezione del nucleo del CMS. Nel nucleo del CMS esistono due puntatori, MAINHIGH e FREELOWE, che contengono rispettivamente l'indirizzo della prima area libera dopo la zona utilizzata dalle GETMAIN e l'indirizzo piu' basso raggiunto dalle DMSFREE. MAINHIGH deve essere sempre minore o uguale a FREELOWE. Una richiesta di memoria che rende invalida questa relazione viene rifiutata con un messaggio di errore e, generalmente, con un ABEND. Il caricamento di programmi rilocabili e non rilocabili e' consentito solo nell'area transiente e nello spazio compreso fra la locazione X'20000' e l'indirizzo contenuto in FREELOWE. Il sistema ha tuttavia la possibilita' di caricare programmi a qualuque indirizzo e, in particolare, in aree precedentemente allocate con una DMSFREE. Questa possibilita' e' sfruttata dal CMS per caricare i programmi del CMSSEG quando non e' possibile utilizzare il segmento "shared".

COMPATIBILITA' CON I SISTEMI OS

Il loader del CMS carica moduli oggetto rilocabili aventi lo stesso formato accettato dall'OS. I moduli oggetto generati dai compilatori e dagli assemblatori dell'OS sono cioe' compatibili con il LOADER del CMS. I moduli oggetto generati dai compilatori OS e i sottoprogrammi di libreria richiamabili da tali moduli possono pero' contenere chiamate al suervisore in cui, ovviamente, si suppone che il sistema operativo sia l'OS. Poiche' il CMS e' in grado di interpretare e simulare correttamente buona parte delle richieste di supervisore dell'OS, l'esecuzione di moduli generati da compilatori e assemblatori dell'OS e' generalmente possibile anche in CMS. Un'importante conseguenza di questo fatto e' la possibilita' di mandare in esecuzione sotto CMS i compilatori e gli assemblatori stessi dell'OS, nonche' alcuni programmi di utilita' dell'OS. In questo paragrafo viene data una breve descrizione del modo in cui il CMS simula le richieste di supervisore dell'OS.

Operazioni di Entrata/Uscita: I metodi di accesso simulati dal CMS sono il QSAM, il BSAM, il BPAM e il BDAM. I nastri magnetici scritti dal CMS sono leggibili in OS e viceversa. Esistono inoltre comandi di CMS per leggere nastri contenenti un "unloaded partitioned data set". I files creati dai metodi di accesso suddetti sui minidischi CMS hanno, ovviamente, formato CMS, ossia sono scritti in blocchi fisici di 800 caratteri e possono avere i soli formati F e V. Sui minidischi CMS non resta traccia, di conseguenza, del bloccaggio visto dal programma. Se, pero', al file viene dato il numero di filemode 4, come record logico del file CMS viene scritto l'intero blocco fisico creato dal programma. Tutte le macro del BPAM sono simulate correttamente, sebbene il catalogo dei files partizionati di CMS sia completamente diverso da quello dei dati residenti sui dischi di OS, e sia invece compatibile con quello dei files creati dal comando MACLIB di CMS. La principale limitazione dovuta alle diversita' di formato nel catalogo sta nell'impossibilita' di inserire informazioni di utente nelle entrate del catalogo stesso. Il BDAM puo' essere usato sui minidischi CMS solo con files con lunghezza di record fissa. Poiche' sui minidischi di CMS la traccia non ha nessun significato, si suppone che essa contenga sempre 255 records. Le chiavi dei records, se esistono, vengono scritte negli ultimi records del file. Questi ultimi records vengono letti in memoria al momento dell'apertura del file. E' possibile, infine, utilizzare in sola lettura il QSAM, il BSAM e il BPAM per leggere da dischi in formato

OS.

Gestione della memoria: le macro GETMAIN e FREEMAIN sono simulate correttamente nel modo cui si e' accennato in precedenza.

Caricamento di programmi: il Loader dell'OS accetta tre tipi di files contenenti codice oggetto rilocabile: moduli oggetto, librerie di moduli oggetto e librerie di "load modules". Il loader del CMS accetta solo i primi due tipi, che corrispondono, rispettivamente, ai files di tipo TEXT e di tipo TXTLIB. Un'ulteriore differenza e' dovuta al fatto che il Loader del CMS legge i moduli usando macro di CMS, per cui, mentre in OS i files di input vengono specificati per mezzo di schede DD, in CMS la ricerca dei programmi su disco segue una logica completamente diversa: tutti i simboli esterni non risolti dal loader provocano la ricerca sui dischi acceduti di un file avente per nome il simbolo esterno e per tipo TEXT; se un tale file esiste, viene caricato in memoria, altrimenti il simbolo esterno viene ricercato sui cataloghi dei files di tipo TXTLIB elencati in precedenza nel comando GLOBAL. Se il simbolo esterno compare nel catalogo di una libreria, tutto il membro cui appartiene viene caricato in memoria. Le macro dell'OS per il caricamento dinamico di programmi (LINK, LOAD, DELETE e XCTL) vengono simulate correttamente dal CMS, benche' provochino l'attivazione del loader del CMS, per cui la ricerca dei moduli da caricare avviene, come si e' detto, in modo diverso e il parametro DCB viene ignorato. I programmi caricati con le macro suddette non devono, ovviamente, oltrepassare l'indirizzo contenuto in FREELOWE. Il CMS usa inoltre particolari accorgimenti per evitare il ricoprimento di aree allocate con la macro GETMAIN. La macro IDENTIFY e' simulata correttamente dal CMS. Impostando un identificatore contenuto nel nucleo del CMS, si puo' fare in modo che le macro LINK, LOAD e XCTL provochino il caricamento in memoria di moduli non rilocabili. Questo metodo e' molto piu' veloce, ma richiede un'estrema cautela nella scelta degli indirizzi dove generare i files di tipo MODULE.

Multiprogrammazione: in CMS non esistono possibilita' di multiprogrammazione, tuttavia le seguenti macro vengono accettate e parzialmente simulate dal CMS: le macro ENQ, DEQ e DETACH vengono completamente ignorate. La macro ATTACH e' eguivalente a una LINK. La macro WAIT pone in stato di attesa la macchina virtuale; non esistendo altri task attivi sulla macchina, l'uscita dallo stato di attesa puo' verificarsi solo se un'interruzione asincrona attiva una procedura contentente la macro POST.

Gestione del tempo: Le macro STIMER e TTIMER sono simulate dal CMS agendo direttamente sull'"interval timer" situato alla locazione virtuale X'50'. Poiche' l'interval timer virtuale viene aggiornato dal CP solo saltuariamente, la precisione ottenibile con queste macro e' alquanto ridotta. A causa della mancanza di multiprogrammazione, non esiste distinzione fra tempo del task e tempo reale, e non e' possibile usare la macro STIMER per entrare in stato di attesa.

STAMPATO PRESSO IL
SERVIZIO TECNOGRAFICO
DEL CNUCE