Arquitetura de Sistemas Operacionais

3ª Edição

Francis Berenger Machado
Luiz Paulo Maia

Estudo de Caso: OpenVMS

LTC

1. Histórico

No final da década de 60, a Digital Equipment Corporation (DEC) anunciou o lançamento de sua linha de computadores com base no processador PDP-11. A linha PDP alcançou grande sucesso comercial na década de 70, permitindo que a Digital se tornasse uma das líderes de mercado ao introduzir o conceito de minicomputadores. A relação preço/desempenho dessas máquinas era bastante superior à dos mainframes que predominavam no cenário da época.

Em 1973, a Digital designou o engenheiro David Cutler para projetar um sistema operacional de tempo real, denominado RSX-11M, para a plataforma PDP-11. Este sistema já possuía conceitos avançados para a época, como sistema de arquivos hierárquicos, utilização da técnica de swapping e um conjunto de ferramentas de apoio a desenvolvimento de sistemas.

Apesar do sucesso, o PDP-11 possuía uma séria limitação na sua capacidade de endereçamento, restrita a 16 bits. Com base neste fato, a Digital investiu na arquitetura VAX (Virtual Address eXtension) de 32 bits, visando oferecer um novo processador com capacidade de endereçamento que satisfizesse seus clientes por um longo período de tempo. Assim, era mais do que necessário desenvolver um sistema operacional para dar suporte a esta arquitetura que podia endereçar aproximadamente 4 bilhões de bytes. Mais uma vez Cutler foi escolhido para liderar este projeto de um sistema que explorasse a capacidade máxima do novo processador. Nascia, então, o VMS (Virtual Memory System), um sistema operacional de tempo compartilhado que aparecia como uma evolução do RSX-11M.

Em 1978, a Digital lançava seu primeiro computador baseado na nova arquitetura, o VAX 11/780, onde o VMS seria o sistema operacional desta plataforma e de todos os demais modelos desta família de processadores. Ao longo dos anos 80, o VMS se consolidou como um sistema operacional de grande sucesso tanto na área comercial quanto no meio acadêmico.

No início dos anos 90, a Digital lançou o Alpha AXP, processador com arquitetura de 64 bits. Uma nova versão do VMS foi desenvolvida para suportar mais esta plataforma. A evolução do VMS permitiu que fossem incorporadas à sua arquitetura características de sistemas abertos, de acordo com padrões de interface do IEEE (Institute of Electrical and Electronics Engineers) e especificações do OSF (Open Software Foundation) e do consórcio X/Open. A partir dessas novas funcionalidades, a Digital rebatizou o sistema como OpenVMS.

2. Características

O OpenVMS é um sistema operacional multiprogramável e multiusuário utilizado em ambientes de processamento comercial e científico que suporta diversos estilos de operação, incluindo tempo compartilhado, tempo real e até mesmo processamento de transações on-line. O sistema foi projetado para operar em plataformas VAX (CISC) e Alpha AXP (RISC) e possui como principais características:

- implementação em ambientes cliente-servidor;
- processamento em modo batch;
- processamento de aplicações em tempo real;
- interface de comandos e windows;
- compatibilidade de códigos fonte interplataformas;
- linguagem de controle sofisticada;

- implementação em sistemas com multiprocessamento simétrico (SMP);
- implementação de sistemas tolerantes à falha;
- funcionalidades de sistemas abertos;
- kernel do sistema operacional suportado por system services.

3. Estrutura do Sistema

A estrutura do OpenVMS consiste em quatro camadas concêntricas, variando da mais privilegiada (kernel) para a menos privilegiada (user). A seguir são apresentados os componentes do sistema associados a cada uma dessas camadas:

- Modo Kernel: subsistema de E/S. subsistema de memória. escalonamento e system services;
- Modo Executive: Record Management Services (RMS);
- Modo Supervisor : Interpretador de comandos (CLI) e run-time library;
- Modo User : utilitários e comandos DCL e programas de usuários.

4. Processo

O processo no OpenVMS pode ser dividido em quatro partes distintas: imagem, contexto de software, contexto de hardware e espaço de endereçamento virtual. Na terminologia da Digital, uma imagem é o resultado da compilação e linkedição de um programa fonte em qualquer linguagem de alto nível, ou seja, é um programa executável. Para que uma imagem possa ser executada, ela necessita de recursos do sistema, como privilégios e quotas. A totalidade desses recursos pode ser entendida como sendo o processo. A imagem é sempre executada no contexto de um processo.

O contexto de software identifica o processo, seu dono, privilégios e quotas de recursos do sistema. Neste ambiente é determinado o que o processo pode ou não fazer, como, por exemplo, o número máximo de arquivos abertos simultaneamente.

O contexto de hardware armazena o conteúdo dos registradores gerais e do registrador de status. Com estas informações é possível que um processo tenha sua execução interrompida temporariamente, retornando posteriormente sem nenhum problema.

O espaço de endereçamento virtual consiste em uma seqüência de endereços que uma imagem pode referenciar no intervalo entre 0 e 2^{32} nos processadores VAX e de 0 a 2^{64} nos Alpha AXP. O mecanismo de memória virtual implementa todo esse espaço de endereçamento utilizando a memória principal e a secundária.

Um processo pode criar outros processos, dependentes ou não de seu criador. Quando depende do seu criador, um processo é definido como subprocesso; no caso de ser independente, é chamado detached. Um subprocesso compartilha quotas com o processo pai. Além disso, caso o processo pai deixe de existir, o subprocesso também é eliminado. Diferentemente do subprocesso, processos detached não dependem da existência permanente do processo criador.

Durante seu ciclo de vida, um processo passa por diferentes estados no sistema. A seguir são relacionados os estados e as mudanças de estados de um processo.

- **Execução (CURrent):** o estado de execução (CUR) indica que o processo está de posse da UCP, sendo executado;
- **Pronto (COMputable):** o estado de pronto (COM) indica que o processo aguarda apenas por uma chance para ser processado, ou seja, é uma espera pelo uso da UCP. Também existe a possibilidade de o processo aguardar pela UCP fora da memória principal. Nesse caso, seu estado é dito computable outswapped (COMO):
- **Espera (Wait):** o estado de espera indica que o processo aguarda por algum evento ou recurso do sistema para continuar sua execução. No OpenVMS, os estados de espera se dividem em subestados, como LEF, CEF, HIB, SUSP, PFW, FPG, COLPG e MWAIT. Do mesmo modo que no estado de pronto, um processo pode permanecer esperando por um evento ou recurso fora da memória principal (outswapped), como LEFO, CEFO, HIBO, SUSPO, PFWO, FPGO, COLPGO e MWAITO. A tabela abaixo ilustra diferentes tipos de estado de espera.

Estado	Descrição
LEF	Espera por um local event flag.
CEF	Espera por um common event flag.
HIB	Processo em hibernação.
SUSP	Processo suspenso.
PFW	Espera devida à ocorrência de page fault.
FPG	Espera por uma página livre.
COLPG	Espera por page fault em pág. compartilhada.
MWAIT	Espera por mutex ou por outro rec. qualquer do sistema.

5. Gerência do Processador

A gerência do processador implementada pelo OpenVMS define a política de divisão do tempo do processador entre os processos dos usuários e do sistema operacional. O escalonamento de processos é realizado por uma rotina de interrupção do sistema denominada scheduler.

A política de escalonamento é implementada através de prioridades associadas aos processos, denominadas prioridades base. Esta prioridade é estabelecida no momento da criação do processo. O OpenVMS implementa 32 níveis de prioridades, divididos em duas faixas: tempo compartilhado (time-sharing) de 0 a 15 e tempo real (real-time) de 16 a 31.

5.1 Escalonamento de Tempo Compartilhado

Um processo em estado corrente (CUR), trabalhando na faixa de tempo compartilhado, somente deixa a UCP caso ocorra uma destas situações:

- término de execução da imagem;
- processo de maior prioridade entra em estado de COM (preempção por prioridade);
- solicitação de um evento ou recurso do sistema;
- término da fatia de tempo (prioridade por tempo).

Para o escalonamento de tempo compartilhado, além da prioridade base definida na criação do processo, existe uma outra, chamada dinâmica, que varia de acordo com as características de execução do processo. O escalonamento de tempo compartilhado é realizado com base na prioridade dinâmica dos processos.

A prioridade dinâmica é alterada quando um processo sai do estado de espera para o estado de pronto. O sistema incrementa um valor à prioridade base em função do tipo de espera a que o processo estava submetido. Eventos que exigem longo tempo de espera incidem em um incremento maior. Com isso, um processo CPU-bound tende a ter uma prioridade dinâmica menor que a de um processo I/O-bound. Este esquema permite balancear o uso do processador entre todos os tipos de processos.

A prioridade dinâmica é calculada pela soma da prioridade base com o incremento recebido. Seu valor é decrementado ao longo do tempo, porém nunca poderá cair abaixo da prioridade base estabelecida.

5.2 Escalonamento de Tempo Real

Um processo em estado corrente (CUR), trabalhando na faixa de tempo real, somente deixa a UCP caso ocorra uma das seguintes situações:

- término de execução da imagem;
- processo de maior prioridade entra em estado de COM (preempção por prioridade);
- solicitação de um evento ou recurso do sistema.

Existem duas diferenças na política de escalonamento para esses tipos de processos: a não-existência do conceito de fatia de tempo e de prioridade dinâmica.

6. Gerência de Memória

A gerência de memória está intimamente ligada à arquitetura do processador. Em função disso existem pequenas diferenças no OpenVMS entre a implementação deste subsistema nos processadores VAX e Alpha AXP. Neste item, a abordagem será relativa apenas à arquitetura VAX.

A arquitetura VAX dispõe de hardware específico para a gerência da memória virtual. Este hardware possibilita que o sistema operacional ofereça uma forma flexível e eficiente na implementação da memória virtual. Podemos destacar como principais características:

- uma lógica de espaço de endereçamento virtual linear;
- mecanismo de proteção da memória;
- compartilhamento de código e dados do sistema operacional;
- mecanismos de tratamento para exceções geradas por referências a páginas não residentes.

6.1 Espaço de Endereçamento Virtual

A arquitetura VAX proporciona um espaço de endereçamento virtual para cada processo de 4 Gbytes. Grande parte desse espaço é utilizada para informações de controle, estando disponível para o programa e dados aproximadamente 1 Gbytes.

O espaço de endereçamento virtual que cada processo possui é um espaço imaginário e contíguo de células. Os endereços referenciados por um programa são sempre virtuais. Em tempo de execução, tais endereços virtuais são transformados em endereços físicos. O subsistema de gerência de memória é responsável por simular a existência do espaço virtual.

A implementação da memória virtual é realizada através de um mecanismo chamado mapeamento. Esta é a maneira pela qual o sistema consegue encontrar a correta localização física (em disco ou memória física) correspondente a um endereço virtual.

A gerência de memória implementa a técnica de paginação, responsável por manter em memória física somente a parte da memória virtual do processo em uso. Nos sistemas VAX, a página é definida como tendo 512 bytes. Com base nisso, tanto a memória virtual quanto a memória física são divididas em blocos de tamanho de 512 endereços. A memória virtual é dividida, então, em blocos de 512 endereços virtuais, denominados páginas virtuais. Da mesma forma, a memória principal é dividida em blocos de 512 células de 1 byte, denominados páginas físicas ou frames.

Fixar em 512 o tamanho da página é o resultado de uma série de ajustes. Entre estes, o fato de que a quantidade mínima de dados que o disco, nos sistemas VAX, transaciona é de 512 bytes (1 bloco). Existem também outros motivos que, por serem inerentes ao mapeamento, serão vistos no decorrer deste assunto.

O conjunto de frames que um processo possui na memória principal em determinado instante é denominado working set. Quando um programa é executado, parte de seu código e/ou dados está no working set, enquanto que o restante é deixado em disco. Embora algumas páginas estejam no working set do processo e, conseqüentemente, na memória principal, elas ainda estão escritas em termos de endereços virtuais e, portanto, suas instruções precisam ser mapeadas antes de serem executadas pela UCP.

Para fazer o mapeamento entre a memória virtual e a memória principal, o sistema utiliza uma estrutura de dados chamada tabela de páginas (page table). Cada página da memória virtual gera um registro na tabela de páginas denominado Page Table Entry (PTE). Toda PTE tem, entre outras informações, um bit que indica se a página associada àquela PTE está ou não no working set do processo (bit de validade).

Sempre que se referencia uma página, o bit de validade é verificado. O fato de ele estar ligado indicará que a página é válida, pois está no working set. Desta forma, o sistema poderá localizar a página na memória real, através das demais informações contidas na PTE. Por outro lado, se o bit de validade não estiver ligado, significará que esta é uma página não-válida, pois

não está no working set. Neste caso, podemos dizer que ocorreu um page fault, que é a exceção gerada quando uma página referenciada não está no working set. Quando isto acontece, o sistema se encarregará de trazer a página para o working set, atualizar a PTE e executar novamente a referência à página, que agora será válida.

6.2 Endereço Virtual

O endereço virtual é estruturado em duas partes, correspondentes à página virtual e ao byte dentro da página. O virtual page number (VPN, bits 9:29) indica a página virtual em que o endereço está contido, e o campo de deslocamento (bits 0:8) indica qual o byte especificado dentro da página.

O espaço de endereçamento virtual de cada processo é dividido em duas regiões denominadas região do processo e região do sistema. O bit mais significativo do endereço virtual (bit 31) indica se o endereço pertence à região do processo (bit50) ou à do sistema (bit51).

A região do sistema é compartilhada por todos os processos. Neste caso, quando um processo faz referência a um endereço virtual dessa região, este tem o mesmo significado para todos os processos. Isto significa que quando dois processos fazem referência a um mesmo endereço virtual da região do sistema, eles estarão referenciando o mesmo código ou dado. Todo código do sistema operacional compartilhado entre processos está localizado nesta região. Para que este compartilhamento seja realizado de forma organizada, o sistema implementa mecanismos de proteção para restringir o acesso às páginas.

Tanto a região do processo quanto a região do sistema são subdivididas em duas regiõ-es, totalizando quatro regiões no espaço de endereçamento virtual. Cada uma dessas sub-regiões representa 1 Gbytes de endereçamento. As duas sub-regiões do espaço do processo são denominadas, respectivamente, região do programa (P0) e região de controle (P1).

Quando se executa um programa do usuário, o módulo executável é alocado em endereços virtuais da região P0, começando no endereço 0 e crescendo para os endereços maiores. Já na região P1, os endereços virtuais são alocados na seqüência dos maiores para os menores, em vista de tal região ser utilizada por estruturas de pilhas.

A região do sistema também é dividida em duas sub-regiões de 1 Gbytes (S0 e S1). Os endereços virtuais da sub-região S0 contêm o código e estruturas do sistema operacio-nal, como rotinas de interrupções, rotinas do sistema e tabelas de páginas. A sub-região S1 é reservada para uso futuro. Atualmente, a UCP não traduz um endereço virtual localizado nesta região. Caso algum processo faça referência a um endereço dentro dessa faixa, um erro de hardware ocorrerá.

6.3 Working Set

O working set de cada processo tem um tamanho máximo que limita o número de frames na memória principal que um processo pode possuir. Esta é uma forma que o sistema implementa para aumentar o grau de compartilhamento da memória entre os diversos processos. Quando este limite é alcançado e o processo necessita trazer uma nova página para a working set, um page fault acontece, e uma página deve ser cedida em troca de uma nova página.

O OpenVMS implementa a política de descartar a página mais antiga presente no working set. Para isso, o sistema mantém uma estrutura chamada lista do working set (working set list) para cada processo. Aparentemente esta política pode não parecer tão boa, se partirmos do princípio de que a página descartada pode ser uma página muito referenciada. Posteriormente entenderemos que o sistema implementa mecanismos que compensam a opção por esta política.

No momento em que uma página é retirada do working set, esta pode tomar rumos diferentes em função das suas características. Para entender como isso ocorre, precisamos antes distinguir as páginas modificáveis das não-modificáveis. As páginas modificáveis são aquelas que contêm dados ou variáveis que naturalmente trocam de valor durante a execução do programa. Por outro lado, as páginas que contêm código não podem ser alteradas, sendo por isso chamadas de não modificáveis. Uma página modificada não pode ser jogada fora quando sai do working set, pois ela contém dados necessários à continuidade da execução do programa. É preciso, então, que essa página, ao sair do working set, seja gravada em disco para que, na próxima referência, a mesma volte com os valores preservados. As páginas não modificadas, por sua vez, não precisam desse tratamento, pois podem ser retiradas do arquivo executável quando necessário.

O tempo de leitura e o de gravação em disco são reconhecidamente grandes quando comparados com o tempo de acesso à memória. Imagine se a cada entrada ou saída de página do working set fossem feitas leitura e gravação em disco. Para contornar esse problema o sistema mantém duas listas que funcionam em conjunto no sentido de dimi-nuir as operações com disco. Essas listas são a lista de páginas livres (free page list) e a lista de páginas modificadas (modified page list), que funcionam como será des-crito.

Além do bit de validade, cada PTE tem um bit de modificação que indica se a página sofreu alteração no seu conteúdo ou não. Assim, quando uma página é retirada do working set para que outra entre, o bit de modificação é verificado. Caso este esteja ligado indicando que houve modificação, a página é passada da lista do working set do processo para a lista de páginas modificadas. Por outro lado, se o bit estiver desligado, a página é passada para a lista de páginas livres.

A lista de páginas livres é responsável por manter informações sobre as páginas físicas de memória que estão disponíveis para uso. Quando uma página é liberada do working set do processo para a lista de páginas livres, esta página não é imediatamente utilizada, já que a página deve percorrer a lista até seu início. Desta forma, as páginas que saem do working set permanecem na memória por algum tempo, permitindo que essas mesmas páginas possam voltar ao working set sem idas ao disco.

Na ocorrência de um page fault, o sistema pode eventualmente encontrar a página requerida na lista de páginas livres ou na lista de páginas modificadas, gerando assim o que é chamado

de page fault barato (page fault sem operação de E/S). Quando não ocorre o page fault barato, o sistema é obrigado a trazer a página requerida do disco, utilizando um novo frame na qual a página será colocada.

A lista de páginas livres fica, eventualmente, com poucas páginas disponíveis para atender a demanda dos processos. Nesse caso, as páginas que estão na lista de páginas modificadas são gravadas em disco (arquivo de paginação) e passadas para a lista de páginas livres. Note que apenas nesse caso as páginas modificadas são gravadas, em conjunto, no disco e, portanto, apenas uma ida ao disco é efetuada.

Quando ocorre um page fault, a página requerida pode estar em vários lugares, como no arquivo executável, na lista de páginas modificadas, na lista de páginas livres ou no arquivo de paginação. O arquivo de paginação é uma área em disco comum a todos os processos, onde as páginas modificáveis são salvas para posterior reutilização.

6.4 Swapping

A paginação é um procedimento natural do mecanismo de memória virtual. Os processos estão permanentemente paginando e, para isso, o OpenVMS deve possuir sempre páginas disponíveis na memória principal para serem utilizadas. Em função disso, a lista de páginas livres deve sempre possuir uma quantidade mínima de páginas necessária para a utilização dos processos.

Em determinadas situações, a memória principal pode estar sendo bastante utilizada, levando a lista de páginas livres a valores baixos. Nesta situação específica, o sistema seleciona um ou mais processos e retira temporariamente o(s) working set(s) da memória principal, gravando-o(s) em disco (arquivo de swap). Com isso, páginas da memória principal são desocupadas, permitindo que o mecanismo da paginação continue seu funcionamento.

7. Sistema de Arquivos

O OpenVMS trabalha com arquivos em um formato próprio, conhecido como Files-11 ODS-2 (On-Disk Structure Level 2). Essa arquitetura de disco define como os dados são armazenados em disco e os arquivos de controle necessários para dar suporte a essa arquitetura.

A menor estrutura lógica endereçável em um disco Files-11 ODS-2 é um bloco, composto por 512 bytes. A alocação de espaço em disco, para a criação e extensão de arquivos, é feita utilizando uma unidade denominada cluster. Um cluster é formado por 1 ou mais blocos contíguos, podendo variar entre 1 a 65.535. O tamanho do cluster é definido durante a inicialização do disco. Um conjunto de clusters contíguos é chamado de extent, podendo um arquivo ser formado por um ou mais extents.

O sistema de arquivos do OpenVMS tem como principais características:

- identificação de arquivos com até 39 caracteres para o nome e mais 39 caracteres para extensão;
- estrutura de diretório em árvore de até oito níveis;
- organização de arquivos seqüencial, relativa e indexada;
- rotinas RMS para operações de E/S;
- proteção por grupo e por lista de controle de acesso;
- alocação de espaço em disco não contígua;

- detecção automática de blocos com defeitos;
- controle de alocação de espaço em disco por usuário.

8. Gerência de Entrada/Saída

A gerência de entrada e saída do OpenVMS é estruturada em camadas, onde as camadas superiores escondem os detalhes das camadas inferiores.

A camada de mais alto nível é o RMS (Record Management Services), que dá suporte a acesso a arquivos e registros. Todas as linguagens de alto nível fazem uso do RMS para executar operações de E/S.

A camada intermediária, composta pelas system services de E/S, é responsável pela interface entre o código do sistema operacional dependente do dispositivo e as aplicações do usuário. As chamadas às system services $QIO permitem executar operações de E/S não suportadas pelo RMS.

O device driver é o código do sistema operacional com características específicas de cada dispositivo conectado ao sistema. Uma aplicação de usuário raramente trabalha neste nível para operações de E/S.