Gerência e escalonamento de processos

Gerência de processos

Um processo é um programa em execução. O status da atividade atual de um processo é representado pelo valor do contador do programa e pelo conteúdo dos registradores do processador. O layout de memória de um processo é tipicamente dividido em várias seções, que incluem:
Seção de texto — o processo, conhecido também como “seção de texto”, inclui a atividade atual, representada pelo valor do contador de programa.
Seção de dados — inclui as variáveis globais do processo.
Seção heap — memória alocada dinamicamente durante o tempo de execução do programa.
Seção pilha (stack) — armazenamento de dados temporário ao invocar funções (como parâmetros de função, endereços de retorno e variáveis locais).

Algo interessante a se considerar é que os tamanhos das seções de texto e dados são fixos, pois não são alterados durante o tempo de execução do programa. No entanto, as seções de pilha e heap podem diminuir e crescer dinamicamente durante a execução do programa. Cada vez que uma função é chamada, um registro de ativação contendo parâmetros de função, variáveis locais e o endereço de retorno é colocado na pilha. Quando o controle é retornado da função, o registro de ativação é retirado da pilha. Da mesma forma, o heap crescerá à medida que a memória for alocada dinamicamente e diminuirá quando a memória for retornada ao sistema. Embora as seções de pilha e heap cresçam uma em direção à outra, o sistema operacional deve garantir que elas não se sobreponham (SILBERSCHATZ; GAGNE; GALVIN, 2012).

Além disso, devemos enfatizar que um programa por si só não é um processo. Um programa é uma entidade passiva, como um arquivo contendo uma lista de instruções armazenadas no disco (geralmente chamada de executável). Em contraste, um processo é uma entidade ativa, com um contador de programa especificando a próxima instrução a ser executada e um conjunto de recursos associados. Um programa se torna um processo quando um arquivo executável é carregado na memória.

Embora dois processos possam estar associados ao mesmo programa, são considerados duas sequências de execução separadas. Por exemplo, vários 2 Componentes de um sistema operacional usuários podem estar executando cópias diferentes do programa de e-mail ou o mesmo usuário pode invocar várias cópias do programa do navegador da web. Cada um desses é um processo separado; e, embora as seções de texto sejam equivalentes, variam as seções de dados, heap e pilha. Também é comum ter um processo que gera outros processos durante a execução (SILBERSCHATZ; GAGNE; GALVIN, 2012).

Conforme um processo é executado, ele muda de estado. O estado de um processo é definido em parte pela atividade atual desse processo. Um processo pode estar em um dos seguintes estados:
Novo — o processo está sendo criado.
Execução — instruções estão sendo executadas.
Espera — o processo está aguardando a ocorrência de algum evento.
Pronto — o processo está aguardando para ser atribuído a um processador.
Terminado — o processo terminou a execução.

Além disso, cada processo é representado no sistema operacional por um bloco de controle de processo (PCB) — também chamado de bloco de Componentes de um sistema operacional 3 controle de tarefas. Ele contém muitas informações associadas a um processo específico, incluindo:

Estado do processo
Contador de programa
Registros da CPU
Informações de agendamento de CPU
Informações de gerenciamento de memória
Informação de Métricas
Informações de status de E/S

Escalonamento de processos

Quando um computador tem suporte a multiprogramação, frequentemente ele tem vários processos competindo pela CPU ao mesmo tempo. Quando mais de um processo está no estado pronto e existe apenas uma CPU disponível, o sistema operacional deve decidir qual deles vai executar primeiro. A parte do sistema operacional que faz essa escolha é chamada de escalonador; o algoritmo que ele utiliza é chamado de algoritmo de escalonamento. O objetivo da multiprogramação é ter algum processo em execução, o tempo todo, para maximizar a utilização da CPU. O objetivo do compartilhamento de tempo é alternar o núcleo da CPU entre os processos com tanta frequência que os usuários podem interagir com cada programa enquanto está em execução. Para atender a esses objetivos, o agendador de processos seleciona um processo disponível (possivelmente de um conjunto de vários processos disponíveis) para a execução do programa em um núcleo. Cada núcleo da CPU pode executar um processo de cada vez. Para um sistema com um único núcleo de CPU, nunca haverá mais de um processo sendo executado por vez, enquanto um sistema multicore pode executar vários processos ao mesmo tempo. Se houver mais processos do que núcleos, os processos em excesso terão que esperar até que um núcleo seja liberado e possa ser remarcado. O número de processos atualmente na memória é conhecido como o grau de multiprogramação (TANENBAUM; BOS, 2014).

Evidentemente, em diferentes ambientes, são necessários diferentes algoritmos de escalonamento. Em outras palavras, o que o escalonador deve otimizar não é a mesma coisa em todos os sistemas. É importante distinguir três ambientes:
1. Lote: Nos sistemas de lote, não há usuários esperando impacientemente uma resposta rápida em seus terminais. Consequentemente, com frequência são aceitáveis algoritmos não-preemptivos ou algoritmos preemptivos com longos períodos para cada processo. Essa estratégia reduz as trocas de processo e, assim, melhora o desempenho.
2. Interativo: Em um ambiente com usuários interativos, a preempção é fundamental para impedir que um processo se aproprie de todo o tempo de CPU e negue serviço para os outros. Mesmo que nenhum processo seja executado para sempre intencionalmente devido a um erro de programa, um único processo poderia barrar os outros indefinidamente. A preempção é necessária para evitar esse comportamento.
3. Tempo real: Nos sistemas com restrições de tempo real, às vezes, a preempção é, estranhamente, desnecessária, pois os processos sabem que não podem ser executados por longos períodos e normalmente fazem seu trabalho e são rapidamente bloqueados. A

diferença com os sistemas interativos é que os sistemas de tempo real executam apenas os programas necessários a uma aplicação em particular. Já os sistemas interativos são de propósito geral e podem executar qualquer tipo de programa, inclusive os que são mal-intencionados.

Objetivos dos algoritmos de escalonamento

Todos os sistemas:
Imparcialidade – dar a cada processo o mesmo tempo de uso de CPU
Imposição da política – garantir que a política declarada é executada
Balanceamento de carga – manter todas as partes do sistema ocupadas

Sistemas de lote:
Taxa de saída – maximizar o número de jobs por hora
Tempo de retorno – minimizar o tempo entre envio e término
Utilização da CPU – manter a CPU ocupada o máximo de tempo possível

Sistemas interativos:
Tempo de resposta – atender rapidamente as requisições
Proporcionalidade —satisfazer às expectativas dos usuários

Sistemas de tempo real:
Cumprir os prazos finais – evitar a perda de dados
Previsibilidade – evitar degradação da qualidade em sistemas multimídia.

Algoritmos de escalonamento em sistemas de lote:

O primeiro a chegar é o primeiro a ser atendido;
Tarefa mais curta primeiro;
Menor tempo de execução restante;
Escalonamento em três níveis;

Algoritmos de escalonamento em sistemas interativos:

Escalonamento round-robin;
Escalonamento por prioridade;
Escalonamento por múltiplas filas;
Processo mais curto em seguida;
Escalonamento garantido;
Escalonamento por sorteio;
Escalonamento com compartilhamento imparcial;

Algoritmos de escalonamento em sistemas de tempo real:

Tempo real rígido (hard real time);
Tempo real relaxado ou não-rígido (soft real time);