Sistema Operacional Solaris

Disciplina: Sistemas Operacionais

Professor(a): Natália Fernandes

Alunos: Amanda Espíndola

Elias Mainetti

Erick Mandarino

Luiza Herback

Niterói, 22 de maio de 2012

SUMÁRIO

Introdução

Antes da invenção do primeiro sistema operacional, os computadores eram compostos basicamente de válvulas e painéis e possuíam dimensões gigantescas. Durante muitos anos seguintes ocorreram várias contribuições para a então construção do sistema operacional como hoje é conhecido.

Existem vários tipos de sistemas operacionais, cada um possui um certo tipo de complexidade e diferem pelos tipos de funções que é provido, porém todos possuem o mesmo compromisso de controlar de dispositivos, gerenciar recursos e manter a integridade do sistema.

O Solaris é um sistema operacional baseado em UNIX com grande inserção no mercado. Possui ferramentas gráficas que permitem um fácil gerenciamento de seus serviços, garantem alta performance e a possibilidade de portabilidade do sistema operacional e dos seus aplicativos.

Neste trabalho, será apresentada a estrutura e o funcionamento do sistema operacional Solaris, bem como a gerência dos seus processos e memória, o escalonamento e a interação de tarefas, sistema de arquivos, além dos seus maiores benefícios frente a outros sistemas operacionais existentes.

Motivação

Solaris é um sistema operacional UNIX desenvolvido pela Sun Microsystems, antiga subsidiária da Oracle.

As raízes do UNIX datam o final da década de 60, quando Ken Thompson do Bell laboratories foi motivado a desenvolver um sistema operacional que desse suporte a programadores em ambiente de pesquisa, dando origem ao primeiro sistema UNIX escrito em assembly em um PDP-7, minicomputador produzido pela Digital Equipment Corporation.

Em 1978, o UNIX foi reescrito em C por Dennis Ritchie, dando origem a sétima versão do sistema. Este foi um dos marcos mais importantes deste sistema operacional, pois tornava possível a portabilidade do sistema para outras arquiteturas com grande facilidade.

Isto ocasionou a criação de várias versões distintas em paralelo, até a fundação da Sun Microsystems em 1982, onde enxergaram a necessidade de unificar todas as versões existentes do sistema UNIX, de modo a criar um sistema aberto padrão, foi onde surgiu o SunOS.

Em 1992 surgiu a primeira versão do Solaris, baseada no System V, que unia as melhores qualidades de todos os sistemas UNIX já desenvolvidos anteriormente, foi criado pela American Telephone & Telegraph (AT&T), e possuía muitas semelhanças com o SunOs.

Hoje, o Solaris está em sua versão 11 e possui benefícios que estão muito acima de outros sistemas operacionais encontrados no mercado, além de possuir ambiente operacional grátis, que possibilita seu uso em qualquer computador sem o pagamento de licença, o Solaris atribui ferramentas para uma fácil expansão do ambiente computacional, tornando-o altamente escalável. Possibilita que operadores façam modificações no sistema sem interromper seu funcionamento, possui também a vantagem de interoperar com sistemas legados e ainda se conectar com novos tipos de aplicações.

Por proporcionar um ambiente estável, de alta perfomance e seguro, o Solaris possui grande força junto ao meio corporativo, sendo adotado para atender empresas de grande porte no mercado e tornando objeto de estudo desta dissertação.

Definindo conceitos

Nos próximos tópicos deste trabalho iremos abordar sobre alguns assuntos básicos relacionados a sistemas operacionais. Por isso é necessário que se tenha um breve conhecimento sobre alguns conceitos que são abordados a todo momento quando estudamos assuntos desse tipo.

Primeiramente definiremos o "cérebro" do computador, o kernel. Ele é responsável pelo gerenciamento da memória, dos processos, dos arquivos e de todos os dispositivos periféricos. De forma resumida, funciona como um organizador de recursos que disponibiliza as ferramentas necessárias para a execução de uma dada tarefa. Tudo isso é feito de maneira segura, de modo a garantir a segurança do sistema operacional. O kernel funciona a todo instante desde o início do sistema operacional e passa-se despercebido pelo usuário comum.

Um programa se caracteriza como uma sequencia de instruções dada ao computador. Quando um programa é desenvolvido, este se converte em uma lista de instruções que serão executadas pelo sistema do computador. Ele pode ser também um conjunto de tarefas, no qual cada uma delas pode ser executada em tempos distintos.

Os processadores do computador são responsáveis por executar as instruções do programa e, em sistema operacionais, quando um programa está sendo executado, este é chamado de processo, e não mais de programa, ou seja, por definição um processo é um programa em execução.

É de suma importância saber também sobre threads. Eles são apenas uma parte do processo. Por sua vez, o processo pode ser formado por vários threads e cada um pode ser executado separadamente.

O Solaris, como outros sistemas Unix, oferece dois modos de operação de threads: thread em modo usuário (modo não privilegiado) e thread em modo kernel

(modo privilegiado). O Solaris utiliza a API Pthreads para criação de novos threads. Abaixo temos figuras que ilustram os dois tipos de threads.

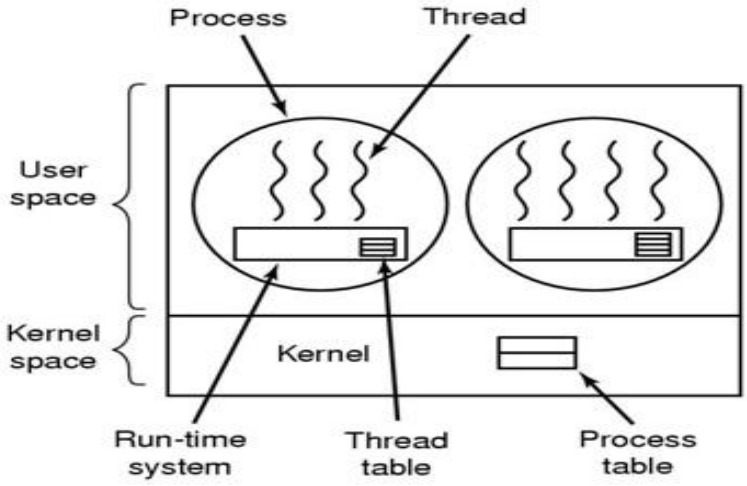

Figura 1 *- Thread em modo usuário*

(Adaptado de Wikipédia)

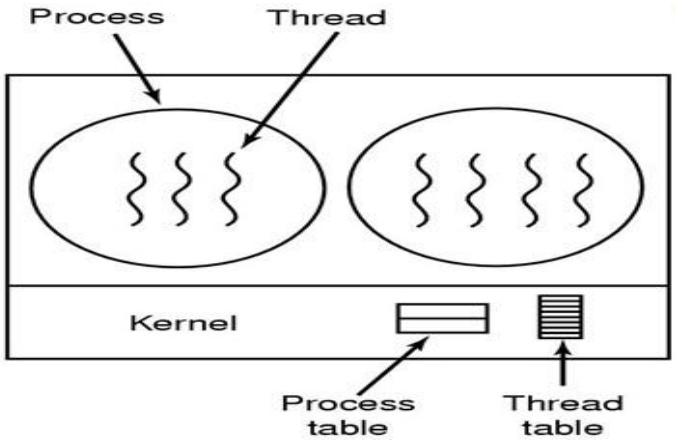

Figura 2 *- Thread em modo kernel*

(Adaptado de Wikipédia)

No modo não-privilegiado, um processo pode acessar apenas sua própria memória, enquanto que no modo privilegiado, o acesso está disponível para todas as estruturas de dados do kernel. O kernel executa processos no modo privilegiado para

evitar que processos de usuário acessem estruturas de dados ou registros de hardware que podem afetar outros processos ou o ambiente operacional. Como as instruções do kernel do Solaris só podem executar em modo privilegiado, o kernel pode mediar o acesso a estruturas de dados do kernel e dispositivos de hardware.

Objetivo e Estrutura do Sistema Operacional

As primeiras versões do UNIX da Sun eram conhecidos como SunOS, que é o nome usado para o componente principal do sistema operacional Solaris. Com o crescimento previsto de sistemas com múltiplos processadores, a Sun investiu fortemente no desenvolvimento de um novo kernel do sistema operacional com um foco primário em escalabilidade com multiprocessadores. O novo kernel permitiu múltiplas threads de execução e forneceu condições para segmentação de processos, o que evoluiu para a escalabilidade encontrada no Solaris hoje. O objetivo da criação do Solaris foi justamente suprir a necessidade de um sistema flexível, com alta escalabilidade, com maior eficiência.

A maioria dos sistemas operacionais modernos utilizam kernel modular, um tipo de estrutura onde o kernel, incluindo todos os seus dispositivos e componentes, formam um único bloco de código mas que agora podem ser compilados independentemente em forma de módulos. A estrutura do sistema operacional Solaris está incluída nesses sistemas baseados em módulos. Um sistema em módulos tem algumas características relevantes, como exemplo, podemos citar: usa uma abordagem orientada a objetos, a comunicação entre os módulos é feita através de interfaces conhecidas, cada módulo é independente do outro e cada módulo é carregado e descarregado de acordo com a necessidade do kernel; este último item caracteriza a flexibilidade desse tipo de estrutura.

O kernel do Solaris é implementado como um conjunto básico de funções do sistema operacional, com os subsistemas do kernel e serviços adicionais ligados em módulos carregáveis dinamicamente. O Solaris é compatível com sete tipos de módulos carregáveis do Kernel: classes de escalonamento, sistemas de arquivos, chamadas de sistema carregáveis, formatos executáveis, módulos streams, drivers de dispositivo e barramento e módulos adicionais diversos. A figura 3 mostra a estrutura modular do Solaris:

Figura 3 - *Estrutura Modular do Solaris*
(Adaptado de SILBERSCHATZ, Abraham; GALVIN, Peter e GAGNE, Greg. Sistemas Operacionais)

Assim como em outros sistemas operacionais com diferentes implementações, o kernel do Solaris também fornece um ambiente de máquina virtual que permite que vários programas executem simultaneamente na plataforma de hardware. Cada programa tem seu próprio ambiente de máquina virtual, com um contexto de execução e de estado.

Um dos recursos do Solaris, é o Solaris Containers, que é uma ferramenta de virtualização do Solaris, que abrange basicamente duas tecnologias: *Solaris Zones*, que implementa o conceito de zonas, cada zona consiste em um ambiente virtualizado que possui sua própria identidade, seu próprio sistema de arquivos, endereço IP, etc, essas zonas ficam separadas da subcamada do hardware ou seja, cada zona comporta-se como se estivesse executando seu próprio sistema operacional possibilitando a execução de cada aplicação em seu próprio ambiente privado, sem que isto afete outros sistemas. Isto oferece uma capacidade de isolamento das aplicações, além de fornecer um mecanismo de controle da distribuição dos recursos entre as diferentes zonas existentes; a segunda tecnologia é o gerenciamento de recursos, que otimiza o processamento através de uma melhor distribuição dos recursos do sistema.

Sistema de arquivos

Os discos rígidos possuem milhões de bits, por isso, é preciso organizar os dados para facilitar a localização e manipulação das informações, este é o objetivo do sistema de arquivos.

O Solaris implementa diversos tipos de arquivos, entre eles podemos citar:

- Arquivos regulares: possuem qualquer dado inserido pelo usuário.
- Diretórios: podem conter arquivos ou outros diretórios.
- Arquivos especiais: estão relacionados à dispositivos de I/O.

O Solaris dispõe de facilidades para armazenamento e gerenciamento de dados. O sistema de arquivos funciona como uma hierarquia de diretórios, formando uma árvore, a partir do diretório raiz, onde um dispositivo pode ser montado sobre um ramo de um sistema de arquivos existente para estender a hierarquia. Há uma única árvore para cada dispositivo de armazenamento.

A vantagem desta disposição em árvore é que o sistema operacional não sabe a localização física dos arquivos, porque não há menção quanto às partes físicas ou discos na árvore. Podemos visualizar abaixo na figura 1 a disposição de um sistema de arquivos:

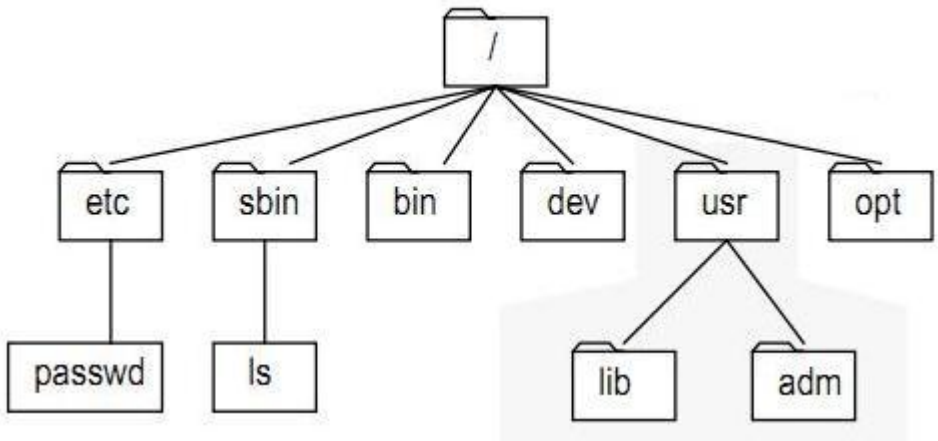

Figura 4 *- Arquivos organizados em uma hierarquia de diretórios*

(Adaptado de MAURO, Jim e MCDOUGALL, Richard. Solaris Internals Core Kernel Architecture)

Solaris fornece um ambiente de suporte onde vários tipos de sistemas de arquivos são executados, entre eles: o sistema de arquivos de disco (como o UFS – Unix File System, MS-DOS File System, CD-ROM File System, ZFS, etc.); o sistema de arquivos virtual (VFS – Virtual File System) e dois sistemas de arquivos baseados em rede, o (NFS –Network File System) e o (RFS – Remote File Sharing).

- *Sistema de arquivos baseados em disco:* UFS (Unix File System) – Sistema de arquivos UNIX, amplamente usado pelo Solaris, principalmente para o disco de boot e ZFS (Zeta Byte File System) – Sistema de arquivos com grande capacidade de armazenamento, que possui segurança de dados avançada e recursos de proteção, eliminando a necessidade de mecanismos de recuperação. Ao redefinir os sistemas de arquivos como o armazenamento virtualizado, o Solaris ZFS permite escalabilidade praticamente ilimitada.
- *Sistema de arquivos virtual:* VFS (Virtual File System) – É uma camada que lida com todas as chamadas de sistema referentes aos sistemas de arquivos, uma interface comum para estes sistemas, que baseiam-se no VFS para interagirem entre si. Gerencia os sistemas que são montados e para isso mantém estruturas de dados que descrevem o sistema de arquivos inteiro (virtual) e os reais, os arquivos. O VFS também mantém um achê de pesquisas de diretório para que os diretórios mais frequentemente utilizados sejam rapidamente encontrados.
- *Sistemas de arquivos baseados em rede:* O NFS (Network File System) funciona como um serviço que realiza o compartilhamento de sistemas de arquivos entre os nós de uma rede. Uma das grandes vantagens da NFS é o seu uso eficiente da largura de banda de rede, conseguido através de RPCs (chamadas de procedimento remoto). Para existir esse compartilhamento de arquivos através da rede, o NFS utiliza uma interface cliente-servidor, algumas características do servidor NFS são: ler e gravar arquivos de acordo com os pedidos de clientes; exportar sistemas de arquivos para disponibilização a futuras aplicações de clientes; não guardar informações de arquivos abertos por clientes; pode ter outros servidores como clientes. E algumas funções do cliente NFS são: pedir ao servidor NFS para ler ou gravar um arquivo; conter informações sobre seus arquivos abertos e montar sistemas de arquivos compartilhados suportados pelo servidor. O RFS (Remote File Sharing) também permite o acesso remoto de arquivos,

porém ao contrário do NFS, o servidor do RFS mantém o estado, para ter controle sobre o número de vezes que um arquivo foi aberto, saber se algum processo bloqueou algum arquivo, etc.

Gerenciamento de memória e escalonamento

O gerenciamento de memória é fundamental para qualquer tipo de sistema operacional. Suas principais funções são: o controle de memória em utilização, a alocação de memória para os processos quando for preciso, a retirada dos dados quando os mesmos forem encerrados e a troca de dados entre a memória secundária e a memória principal quando a última não conseguir armazenar os processos.

O Solaris é um sistema preemptivo, ou seja, o tempo é compartilhado entre os processos a serem executados e, é de responsabilidade do kernel o gerenciamento de memória. Nesse tipo de sistema, cada processo tem um tempo distinto de execução e fica numa fila de espera na memória aguardando sua vez. Vale lembrar que até o fim da execução de um processo ele passa pela CPU inúmeras vezes, ou seja, ele volta à fila de memória repentinamente até ser todo executado. Para isso, é necessário que o gerenciador de memória suspenda ou retorne um processo sempre que preciso. É importante ressaltar que o gerenciador de memória sabe lidar com os processos que estão no aguardo de um dado de entrada e saída para que esses não partilhem o tempo de execução.

A técnica de escalonamento de memória do Solaris segue a ideia de que processos que forem levar muito tempo para serem executados terão níveis de prioridade menores. As tarefas que exigem um tempo maior de execução serão mais lentas não só por seu tempo de execução maior, mas também por terem nível de prioridade mais baixo. Desse modo, um número maior de tarefas “simples” poderão ser utilizadas rapidamente pelo usuário, aprimorando a aparência de processamento em tempo real (*real-time*). Isso tudo é uma questão de escolha dos projetistas que adotam certos caminhos como prioridade.

Por tratar-se de um sistema multithread, esse sistema operacional possui threads especiais no kernel que tratam do caso das interrupções. Essas estão sempre em maior prioridade no sistema e podem usar regras padrões do kernel para bloquear algum

recurso quando for preciso. Nesse sistema operacional os threads possuem duas prioridades, uma que provém da sua classe de escalonamento e outra que é herdada da sua interação com outros objetos – a de escalonamento é sempre maior que as herdadas. Quando um thread de prioridade mais alta estiver bloqueando um recurso, ele poderá transferir temporariamente sua prioridade para um thread de prioridade mais baixa, tal feito é conhecido como inversão de prioridade. Por exemplo, quando um processo de mais baixa prioridade adquire um recurso necessário ao thread de mais alta prioridade, há uma inversão de prioridades, de modo que, o de mais baixa prioridade assume rapidamente uma prioridade de mais alto nível para que ele posteriormente passe o recurso necessário ao thread de mais alto nível de prioridade por escalonamento. A grosso modo, é como se um thread de mais baixo nível assumisse temporariamente o lugar de um thread de mais alto nível, quando for preciso. A figura abaixo ilustra melhor o processo, vejamos:

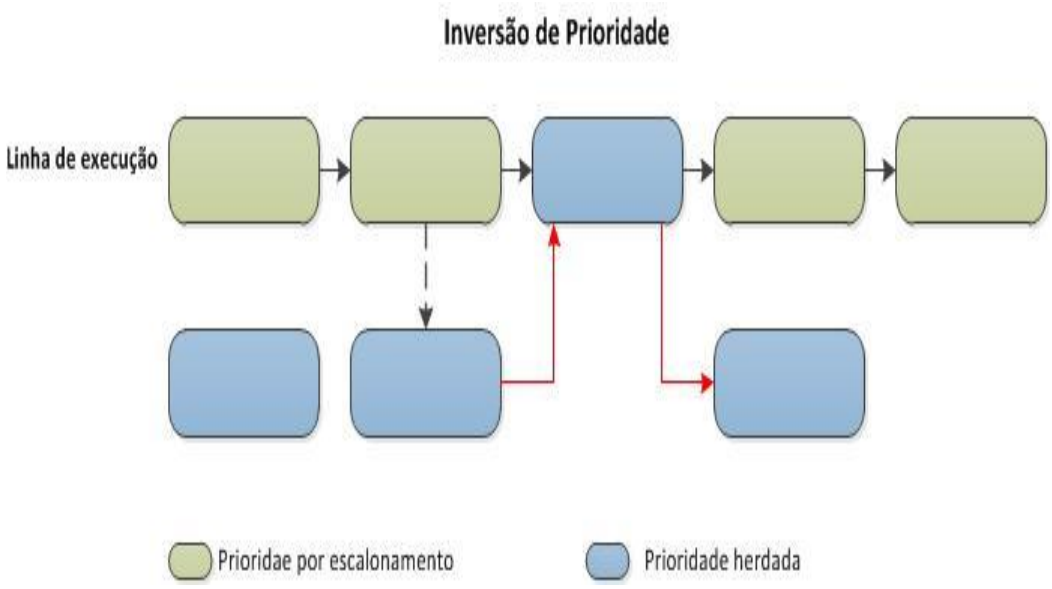

Figura 5 *– Inversão de Prioridade*

A herança de prioridade possui algumas limitações e não é de qualquer forma que pode ser utilizada. Quando temos certeza que o thread irá liberar o recurso

necessário, por exemplo, essa poderá ser implementada. Isso pode ser realizado quando o recurso for mantido por uma única thread.

Para os recursos de sincronização o Solaris possui quatro: mutexes, semáforos, variáveis de condições e bloqueios de leitores/escritores. No caso do mutexes, o thread dono é sempre conhecido e, então, pode se utilizar a técnica de inversão de prioridade. No entanto, semáforos e variáveis de condição não apresentam na maioria das vezes donos, inviabilizando a utilização da herança de prioridade. Por fim, bloqueio de leitor-escritor também permite que se utilize a inversão de prioridade. Nessa última técnica, pode existir diversos "escritores" donos, porém a thread que só herdará a prioridade do "escritor" de mais alta prioridade.

Na computação a limitação do hardware sempre está presente e, um bom exemplo disso, é a limitação da memória principal nos computadores. Dada essa limitação, o Solaris enfrenta esse problema com a técnica de *swapping*, uma espécie de gerência de memória virtual, no qual processos que estão em menor utilização na memória quando ela está cheia, são paginados e colocados na memória secundária. Tais páginas são trazidas de volta a memória principal somente quando forem referenciadas. A vantagem disso é que a paginação ocupada um espaço de memória principal muito menor do que todo um programa carregado nela, já que a página direciona à memória secundária o segmento de dados que antes ocupava a memória principal. Um algoritmo fica analisando toda a memória principal em intervalos de tempo, para caso seja necessário uma nova página, uma atualização, ou exclusão. A priori, todas as páginas são marcadas como não utilizadas (bit de acesso igual a zero), assumindo valor de bit igual a um quando passam a ser referenciadas. Elas são observadas também de tempo em tempo pelo sistema para verificar-se seu bit de acesso. Caso esse tenha se alterado, ocorre uma atualização informando se estão liberadas ou não para uso. Abaixo vemos uma ilustração da técnica de *swapping*.

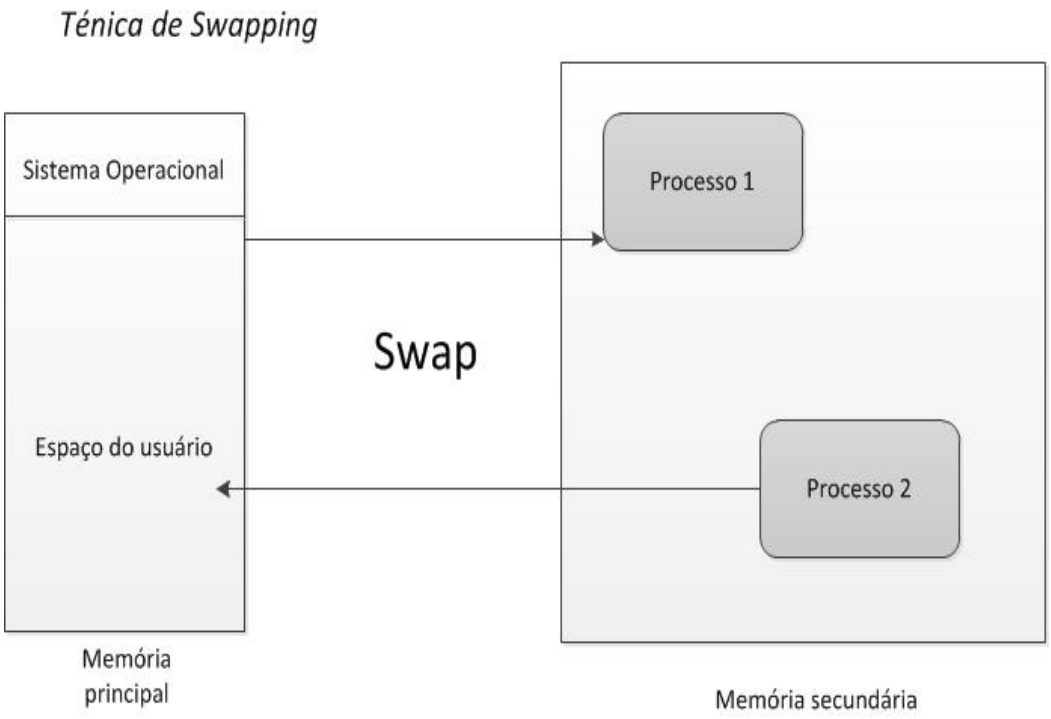

Figura 6 *– Swapping*

Por essas e diversas outras técnicas de gerenciamento de memória e escalonamento, o Solaris torna-se um ótimo sistema multithread e de processamento em tempo real, tanto em monoprocessadores quanto em multiprocessadores.

Programas do Sistema

O Solaris passou a ser mais conhecido principalmente por sua acessibilidade e capacidade de resolução de alguns problemas de forma simples e aprimorada. Tais funcionalidades são possíveis porque ele traz alguns programas de sistema inovadores que facilitam isso, dentre os quais podemos citar como os principais o *Dtrace*, o *Containers*, *ZFS* e o *Self Healing*.

O Dtrace é uma ferramenta poderosa que garante uma visão de tudo que acontece no sistema em tempo real. Isso melhora o diagnóstico de problemas, reduzindo o tempo de correções das falhas. O Containers permite que se crie vários ambientes, cada um com sua própria identidade, no qual cada um simula seu próprio hardware.

Esse recurso é de ótima importância para aplicação em servidores, onde podem existir diversas aplicações rodando simultaneamente, cada uma contando com seus próprios recursos. Por sua vez, o ZFS auxilia no gerenciamento de arquivos, fornecendo ao usuário principalmente ferramentas que solucionam problemas com armazenamento, corrupção de dados e sistemas de arquivos. Por fim, Self Healing faz com que o sistema operacional tenha a capacidade de diagnosticar antecipadamente uma possível falha que possa vir a ocorrer, prevenindo-se de paradas críticas futuras. Quando isso ocorre, automaticamente há um isolamento do processo e/ou hardware que gerou o problema até que seja possível a recuperação do mesmo. Agindo dessa forma, é possível que outros serviços primordiais para o sistema possam continuar operando de forma initerrupta, já que é possível prever antecipadamente um erro e começar a tratá-lo, ou isolá-lo, antes dele ocorrer.

Além disso, o Solaris conta ainda com atualizações automáticas através do *Sun Update Connection* e milhares de outras aplicações provenientes do *Integrated Open Source Applications.*

API de Chamadas do Sistema

Se um processo precisa acessar serviços do kernel do sistema, é preciso fazer a transição do processo do modo usuário para modo kernel, a fim de se obter proteção do uso incorreto de recursos. Essa transição é feita através de um conjunto de interfaces conhecidas como chamadas de sistema. Chamadas de sistema são solicitadas a fim de ter o kernel executando uma função específica (por exemplo, abrir um arquivo) em nome do thread de chamada. Fazem parte das interfaces de programação de aplicativo (APIs) que acompanham o sistema operacional, que são um conjunto de rotinas que acessam ou executam determinadas funcionalidades nos sistemas operacionais. São geralmente usadas por programadores para desenvolvimento de softwares a fim de que não seja preciso envolver-se em detalhes de implementação dos sistemas operacionais.

Uma API podem incluir especificações de rotinas, estruturas de dados, classes de objetos e variáveis. Uma especificação API pode assumir muitas formas, incluindo uma norma internacional, como a POSIX, usada em sistemas como o Solaris. A figura 7 exemplifica o funcionamento de uma chamada de sistema:

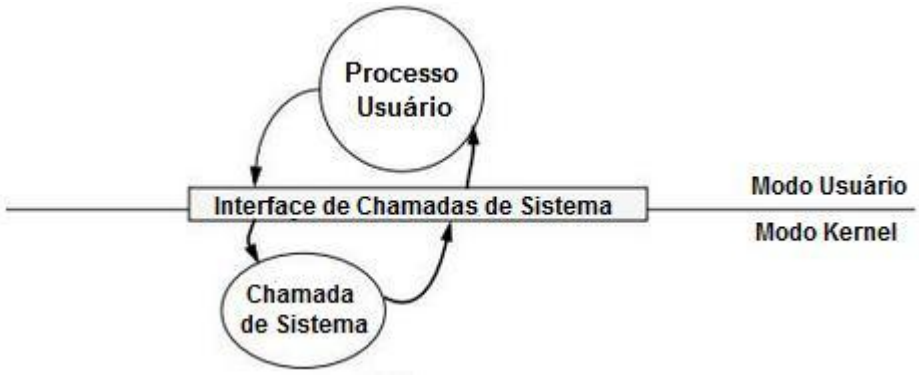

Figura 7 – *Transição entre modos usuário e kernel através de uma chamada de sistema*

(Adaptado de MAURO, Jim e MCDOUGALL, Richard. Solaris Internals Core Kernel Architecture)

A sobrecarga de chamadas do sistema é um problema, pois torna a transição entre os modos lenta, o que deixa a CPU ociosa. Em alguns casos, queremos ser capazes de ter acesso rápido e otimizar as solicitações dos processos. O kernel do Solaris fornece uma estrutura de chamada de sistema rápida para que os processos de usuário possam mudar para o modo kernel de forma protegida para fazer o processamento mínimo e, em seguida, retornar, sem a sobrecarga do quadro completo de chamadas de sistema. Esta estrutura pode ser utilizada apenas quando o processamento necessário no núcleo não interferir significativamente com registros e pilhas. Assim, a chamada do sistema rápida não precisa salvar todo o estado que uma chamada de sistema regular faz antes de executar as funções necessárias.

Gerenciamento de Processos

O kernel do Solaris fornece suporte para uma entidade conhecida como processo, que basicamente é um programa em execução.

O modelo de processo no Solaris é em alguns aspectos semelhante ao modelo tradicional de processo Unix. Para criar um novo processo, a função *fork* é usada. Esta cria um processo filho que é exatamente a cópia do processo pai. Um processo-pai pode ter vários processos-filhos e estes também podem ter seus processos-filhos.

O Solaris é um sistema operacional multiprogramável, ou seja, um sistema que permite que cada usuário tenha vários processos executando simultaneamente. Além disso, ele possui estruturas básicas para lidar com processos, como por exemplo, o

kernel mantém uma tabela de processos em todo o sistema, em que cada processo é identificado por um número inteiro positivo chamado número de identificação do processo (pid), esta tabela também possui todas as informações referentes aos processos, como seu modo, tipo de prioridade, etc.

O gerenciamento de processos é feito pelo kernel do Solaris e é utilizado para mostrar os processos que estão sendo executados em uma determinada estação de trabalho. Além disso, o kernel tem o poder de criar, interromper e reinicializar processos, investigar e depurar processos irregulares (processos que não estão realizando o trabalho esperado). O kernel como gerenciador, também utiliza semáforos como mecanismo de exclusão mútua e escalonamento circular com filas múltiplas, que servem para indicar a prioridade de cada processo.

O gerenciador também controla a comunicação entre processos, que pode ser feita através de memória compartilhada ou por transmissão de mensagens. Na memória compartilhada, os processos criam um segmento onde poderão trocar informações lendo e gravando dados nessa região. Na transmissão de mensagens, os processos se comunicam sem dividir o mesmo espaço de endereço, podendo ser sistemas distribuídos, comunicando-se por uma rede.

Segurança do Solaris

A segurança do Solaris é divida em quatro níveis de proteção, sendo eles:

a) Controle de login

Através dessa ferramenta o administrador controla o acesso dos usuários ao sistema. Sempre que um usuário tenta acessar o sistema ele precisa efetuar um log in com uma senha que permite a identificação do requisitante. Depois de efetuado o log in, o usuário tem acesso aos recursos disponíveis para ele.

b) Controle de acesso aos recursos do sistema

Com o controle de acesso o administrador manipula o acesso geral aos recursos do sistema, provendo também ferramentas de segurança, o administrador também conta com um recurso que permite rastrear as tentativas de acesso.

c) Segurança para desenvolvimento e distribuição de serviços

Neste nível existe a possibilidade de controle dos privilégios de acesso a arquivos de acordo com o usuário, para isso, são utilizados os serviços de autenticação, autorização, privacidade e integridade de dados.

d) Controle de acesso à rede física

O controle de acesso à rede física evita que os usuários exponham dados ou informações importantes acidentalmente através do Solstice Firewall, este produto é responsável pela proteção de dados na rede Intranet, provendo um sistema de registro e alertas contra tentativas de violação.

Uma das grandes vantagens do Solaris está na grande estabilidade do sistema oferecido, isso traz à instituição que o utilza um nível de segurança bastante alto.

O Banco do Brasil usa a plataforma Solaris para gerenciar as redes com a ferramenta HP Open View. Esse programa proporciona um gerenciamento integrado de redes e sistemas em ambientes distribuídos. Através de traps enviados do equipamento monitorado à estação coletora é possível monitorar o estado do equipamento, permitindo uma ação corretora imediata caso ocorra algum problema. Já a estação coletora envia pollings no sentido do equipamento, estes checam se o equipamento está on line.

Através de um range de IP´s o HP Open View cria um mapa de todos os equipamentos que pertencem a agência. O operador responsável pela manutenção dessa rede consegue visualizar o mapa de qualquer estação de trabalho, dessa forma se consegue monitorar o estado dos equipamentos em tempo real. E isso tudo só pe possível graças à estabilidade oferecida pelo Solaris.

Conclusão

Neste trabalho vimos que o Solaris é um sistema operacional de alto potencial, principalmente quando se fala em trabalho em tempo real e escalabilidade. Ele possui diversas ferramentas que aprimoram o seu desempenho e por isso vem sendo utilizado por empresas, já que garante ótimos recursos para se trabalhar com computação de redes, além de contar com recursos de segurança avançados.

Trata-se de um sistema com Kernel dividido em blocos, no qual cada estrutura pode executar sua tarefa separadamente, não precisando acessar todas as outras para executar uma dada operação. Isso faz dele, um dos sistemas atuais de mais alto nível.

Todo o sistema é baseado na plataforma UNIX, o que garante a ele uma alta estabilidade. Suas técnicas de escalonamento e gerenciamento de memória possui ótimos recursos, tais como: inversão de prioridade e swapping, que garantem ótimo manuseio de execução em tempo real no momento de operação dos processos.

O Solaris dispõe de facilidades para o gerenciamento de arquivos, fornecendo um ambiente de suporte onde vários tipos de sistemas de arquivos são executados, entre eles um de grande capacidade de armazenamento, o ZFS, o que aumenta a escalabilidade deste sistema.

Por esses e outros motivos, nos últimos anos tem se tornado uma ótima opção de sistema operacional para se trabalhar com altas tarefas e execuções. Prova disso, é a utilização desse sistema nos servidores de grandes empresas.

Podemos perceber então, a relevância deste sistema operacional e confirmar sua tendência de expansão na computação.

Bibliografia

- MAURO, Jim e MCDOUGALL, Richard. 2000. *Solaris Internals Core Kernel Architecture*.
- SILBERSCHATZ, Abraham; GALVIN, Peter e GAGNE, Greg. 1999. *Sistemas Operacionais.*
- Wikipédia.