UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA

INF01151 – Sistemas Operacionais II N

Grupo 1 - Sistemas Operacionais Distribuídos

(Mach) César

Q: Discorra brevemente sobre a evolução do Sistema Operacional Mach. Não esqueça de mencionar aspectos técnicos importantes como:

a) Mach é um SisOp para qual tipo de sistemas (single/multi-core, distribuído)?

b) Softwares de outros SisOps rodam em Mach? Através de qual mecanismo?

c) Como é o kernel do Mach (monolítico, micro-kernel)?

d) Quais SisOps se baseiam, de alguma forma, com o Mach?

Conclua sua resposta citando os objetivos principais do SisOp em questão.

R: As origens do Sistema Operacional Mach remetem ao Sistema Operacional RIG, o qual começou a ser desenvolvido na Universidade de Rochester no ano de 1975, e ao Sistema Operacional Accent, iniciado em 1979 na Universidade de Carnegie-Mellon. Em ambas ocasiões, é importante salientar o nome de um dos designers dos dois últimos sistemas: Richard Rashid.

O Mach é um S.O. tanto para sistemas single e multi-core como para sistemas distribuídos. Apesar de prover bons mecanismos para comunicação em rede, o objetivo inicial não era se fazer um Sistema Operacional distribuído.

Um ponto interessante deste S.O. é a possibilidade de se emular outros sistemas operacionais na camada do usuário. Essa feature veio da observação feita por Rashid a respeito do Mach estar perdendo espaço para o UNIX em 1984. Ainda que a idéia de emular vários SisOps possa induzir o pensamento de termos um kernel pequeno e modular, até 1988 o kernel do Mach era bastante grande e monolítico (muitos trechos de código do kernel do UNIX estavam presentes nele). Apenas após a versão 2.5 é que se fez a mudança da parte de emulação para o espaço do usuário e o restante do código transformou-se em um microkernel.

Pode-se dizer que o Mac OS X e o iOS possuem forte influência do Mach: Os sistemas operacionais XNU e Darwin, que são a base destes dois sistemas da Apple, foram fortemente baseados no Mach (ainda que uma característica importante como a presença de microkernel não seja tenha sido modificada para um kernel híbrido nos S.O. da Apple).

Os principais objetivos do Mach se relacionam tanto com pesquisa como com desenvolvimento e são:

1) Prover uma base para desenvolver outros SisOps;

2) Suportar espaços de endereçamento grandes e esparsos;

3) Explorar paralelismo tanto no sistema como nas aplicações;

4) Ser portável para uma grande quantidade de máquinas.

(Amoeba) Hugo

Q: O qué são e como se relacionam os conceitos de objeto e capabilitiy no SO distribuído Amoeba?

R:O objeto é um tipo de dados abstrato encapsulado, na qual certas operações bem definidas podem ser executadas. Não contém processos nem métodos.No contexto deste sistema distribuído, o objeto é o conceito básico unificador de todos os serviços e processos Amoeba.

Estes objetos são nomeados e protegidos de una maneira uniforme por tíquetes chamados de capabilities. Para criar um objeto, um cliente faz uma RPC com o servidor apropriado especificando o que quer. O servidor cria o objeto e retorna uma capability para o cliente. Nas operações seguintes, o cliente tem que presentar dita capability para identificar o objeto. Esta capability é um numero binário longo. Para proteger o objeto, a capability contém direitos que indicam quais operações podem ser executadas no objeto. Estas capabilities são protegidas criptográficamente usando funções one-way, cada uma delas contém um campo checksum que garante a segurança da capability.

Q:Qual a diferença entre a criação de processos Unix e processos Amoeba?

R: Os processos Unix são criados duplicando o processo pai (FORK) e dita copia substitui-la quase imediatamente com uma nova copia de código(EXEC). Em um sistema distribuído esse processo de copia/substituição pode resultar muito ineficiente. Em Amoeba, os processo filhos são criados em um processador específico com a sua correspondente imagem de memória desde o inicio mesmo de criação. Isto, se parece muito com a maneira na qual o MS-DOS carrega o seus processos, com a grande diferença que caso do Amoeba, o processo pai pode executar em paralelo enquanto acontece a criação dos seus processos filhos. Isto permite a criação de um número arbitrário de processos filhos, a onde estes processos filhos podem ter seus próprios filhos, levando em uma árvore de processos.

(Chorus) Glauber

Q: Ao passo que o sistema Chorus deixou de ser uma pesquisa acadêmica em sistemas distribuídos e tornou-se um produto comercial os objetivos do sistema evoluíram. A partir disso podemos dizer que Chorus possui 4 objetivos principais. Quais são?

R: Emulação de sistemas UNIX em alta performance; usável em sistemas distribuídos onde o Chorus pretende permitir que programas UNIX possam ser executados em uma coleção de máquinas conectadas por uma rede; suporte a aplicações de tempo real e a integração de programação orientada a objetos.

(DCE) Guilherme

Q: Qual a principal diferença entre o Distributed Computing Environment (DCE) e os sistemas Amoeba, Mach e Chorus?

R: A principal diferença é que o DCE provê um ambiente que serve como plataforma para rodar aplicações distribuídas que roda sobre um sistema operacional existente (UNIX, Windows, VMS e OS/2) enquanto os outros sistemas são um sistema operacional em si. Isso implica que a maior parte dos pacotes do DCE rodem no espaço de usuário e não no kernel.

(Inferno) Débora

Q: Como funciona e para que serve o espaço de nomes do sistema Inferno?

R: A visão que o sistema tem da rede é como se ela fosse um espaço de nomes único e coerente (“The Inferno Namespace”) que aparece como um sistema de arquivos hierárquico, mas que pode representar recursos fisicamente distantes. Ou seja, os recursos são representados de forma uniforme e as aplicações podem usá-los de maneira totalmente transparente. Exemplo de uso: Queremos debugar um programa e para isso precisamos de acesso à pasta */prog*. Esta pode ser local ou remota; se for remota, importamos o espaço de nomes */prog.* Exemplo de importação: *mount tcp!143.107.45.20 /n/remote/prog;* Após, fazemos um bind com o nosso diretório local: *bind /n/remote/prog /homework/prog.*

(Locus e Medusa - Outros SisOps) Anderson

Q:(LOCUS) Qual as características do sistema LOCUS? Escolha a mais notável e explique-a. Hoje em dia quais dessas características ainda são usadas?

R : LOCUS foi um sistema operacional distribuído desenvolvido pela Universidade da Califórnia durante os anos 80. Sua maior preocupação era passar a noção de parecer "uma única máquina" mesmo sendo composta de diversas outras. Nesse enfoque, ele suportava acesso a dados de forma transparente através de um sistema de arquivos bem avançado (usava a mesma interface do Unix, mas recebia como parâmetro quantas cópias deviam ser produzidas e onde o arquivo deveria ser salvo). Oferecia acesso rápido e confiável ao seu sistema de arquivos usando replicação. Assegurava acesso atualizado a seus dados através de um coordenador (chamado CSS - current syncronization site), este era responsável por uma política de sincronização global de acesso aos recursos compartilhados. Ainda, arquivos que precisavam ser diferentes entre si, eram colocados em diretórios ocultos onde o sistema se encarregaria de garantir o acesso atualizado. Desse modo, o sistema distribuído dava a falsa sensação de um comportamento de sistema isolado

Oferecia acesso remoto a dispositivos de I/O bem como a criação de processos remotos. Tinha intensos protocolos para reconfiguração dinâmica de modo a oferecer alta disponibilidade de acordo com os padrões da época. Por seus arquivos serem dispersos entre os diversos componentes do sistema, precisava de mecanismos para coerência de memória que ficaram a cargo do sistema operacional,

A característica mais notável era sua abstração de máquina distribuída para uma máquina singular. Entre seus diversos protocolos, os de alto nível ofereciam transitividade, ou seja, se o local A pode se comunicar com o local B e B pode se comunicar com C, logo A poderia se comunicar com C. Se o local B falhasse então um substituto seria encontrado pelo sistema de modo a não quebrar a relação existente, ou seja, dinamicamente a configuração do sistema podia se modificar de modo a assegurar rapidez ou manutenção diante falhas.
LOCUS usava o conceito de pipes e pipes nomeados ainda usados atualmente entre outros conceitos importantes como criação de processos remotos, traps e RPC.

(Objetivo, Kernel, Conceitos) Diogo

Q: Quais são os objetivos dos sistemas Amoeba, Chorus e Mach(baseados em microkernel)?

R: Amoeba: Criar um sistema operacional distribuido transparente e fornecer um sistema para fazer testes para programação paralela e distribuida

Chorus: alta performance na emulação do UNIX, usar em sistemas distribuídos, integrar com programação OO, aplicações em tempo real.

Mach: Fornecer uma base para para criar outros sisop's, permitir acesso transparente a recursos da rede, explorar paralelismo na aplicação e no sistema, ser portável em várias máquinas.

(Memória) Lucas

Q: Comparando as implementações de memória compartilhada distribuída nos sistemas Amoeba, Mach e Chorus, quais as desvantagens de cada um? Por quê?

R: Amoeba: A memória compartilhada distribuída é obtida através do replicamento dos objetos compartilhados em todas as máquinas que usam eles. Isso causa como desvantagem um update da memória mais custoso.

Mach e Chorus: Nesses sistemas existe potencial de ocorrer thrashing. Isso ocorre pois ambos usam uma memória compartilhada distribuída baseada em páginas. Quando um processo referencia uma página que não está presente em sua máquina, a página é buscada na máquina que a possui e é trazida para a dele. O thrashing pode ocorrer quando duas máquinas acessam muitas vezes a mesma página.

(File system) André

Q: O Sistema Distribuido Amoeba, por exemplo, utiliza 2 modulos separados chamados de AFS (Atomic File System) e DNS (Directory and Name Service). Diga como é feito a localização de um arquivo num sistema que utiliza esse tipo de modularização.

R: O AFS apenas guarda arquivos referenciados por um número. O DNS cuida da gerência de Pastas que indicam os nºs dos arquivos que cada pasta contem. Assim, o DNS resolve caminhos requisitados e entrega o nº do arquivo correspondente, o qual é enviado ao AFS para acesso.

(Comunicação, IPC, RCP, mensagens, Memória compartilhada) Anellena

Q: Que tipo de comunicação cada sistema estudado (Mach, Amoeba, Chorus e DCE) suporta? Explique resumidamente cada uma.

R: Amoeba suporta duas formas de comunicação, RPC, usando passagem de mensagem síncrona ponto a ponto (ou seja, o envio de uma mensagem bloqueia quem enviou até que haja uma resposta) e comunicação em grupo (de processos que estão cooperando para realizar alguma tarefa ou serviço), esses grupo são fechados e a única maneira de algum cliente de fora se comunicar com eles é por RPC e isso foi feito buscando maior transparência nas comunicações internas do grupo. Já Mach suporta uma variedade maior de formas de comunicação, como passagem de mensagens assíncronas, RPC e byte streams, mas a base de tudo isso é uma estrutura de dados do kernel chamada porta, que é, essencialmente, uma mailbox protegida que permite comunicação unidirecional. Chorus suporta comunicações baseadas em passagem de mensagens, usando dois tipos de operações, envio assíncrono (e

sem garantias) e RPC. Como DCE é baseado no modelo cliente/servido, ele suporta apenas RPC.

(I/O) José Luis

Q: Como um processo ou thread pode fazer I/O nos sistemas Amoeba, Mach e Chorus?

R: Amoeba: Fazendo uma RPC para o microkernel, que vai encarregar uma de suas threads do I/O.

Mach: O processo se comunica com os drivers de I/O através do mecanismo chamado de porta, uma espécie de caixa de correio protegida. Quando o processo cria uma porta, um espaço de armazento do microkernel é dedicado àquela porta até sua destruição.

Chorus: O processo faz uma requisição de I/O com uma trap. Quando ocorre uma interrupção para um dispositivo, o kernel espontaneamente cria uma thread para tratá-la. Se outra interrupção ocorre enquanto a primeira ainda não terminou de ser tratada, o kernel cria outra thread.