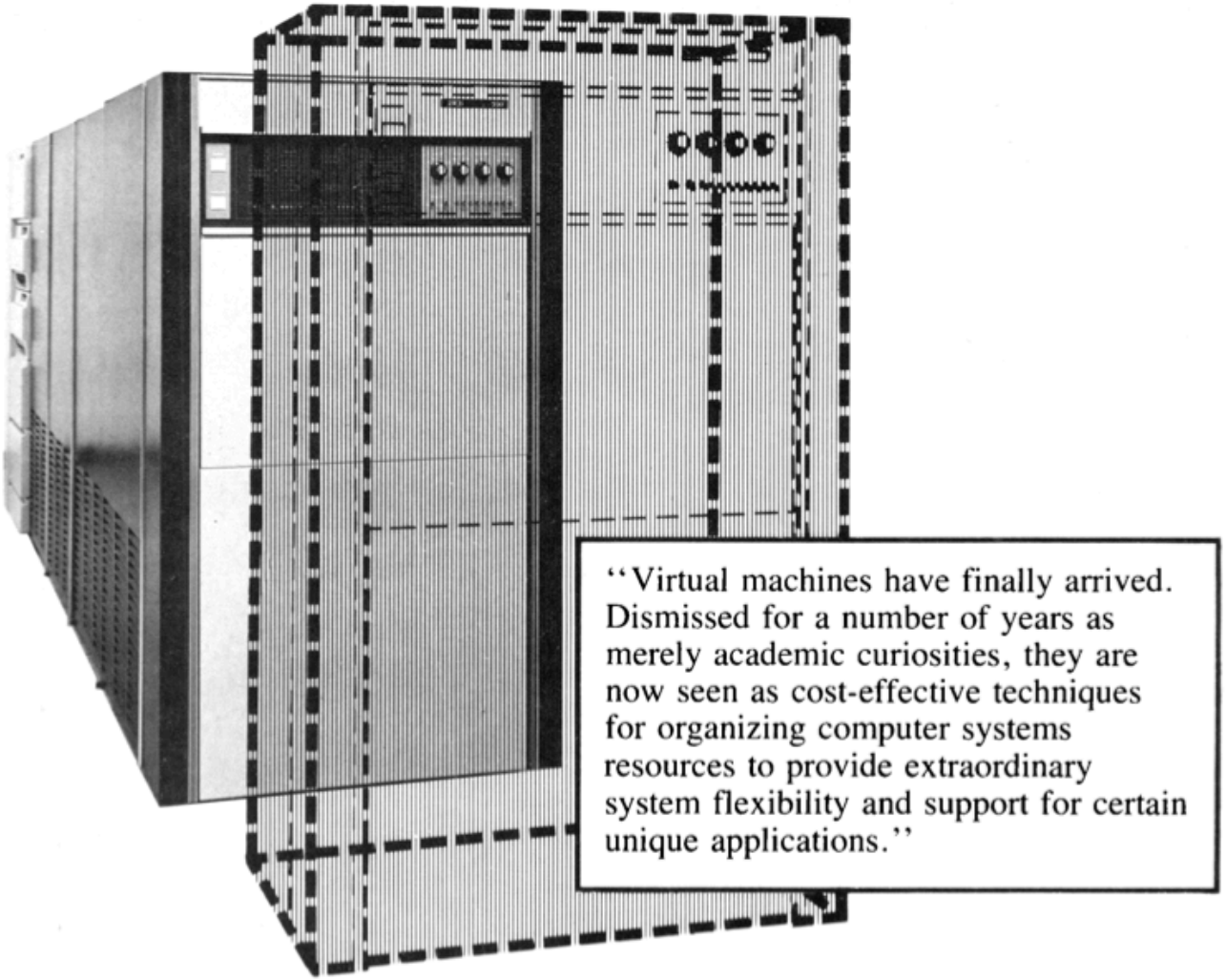
MC714

Sistemas Distribuídos

2° semestre, 2013

Alguns conceitos em máquinas virtuais

Máquinas virtuais



Máquinas virtuais

- Simulação instrução por instrução: conhecida há décadas.
  - Ex.: aplicação para um computador X cujo hardware ainda está em desenvolvimento.
  - Simulador para X (processador, memória, periféricos) que rode em máquina de propósito geral G.
  - Programas que rodam em G poderão rodar em X com mesmos resultados.
  - Espaço de memória simulado, dispositivos simulados, executar instruções na máquina simulada.
  - Camada que filtra e protege os recursos da máquina G.

Máquinas virtuais

- Para múltiplos programas, pode-se:
  - Múltiplas cópias do simulador.
  - Simulador capaz de dividir o tempo entre aplicações.
  - Ilusão de múltiplas cópias da interface de hardware-software da máquina X em G.

Máquinas virtuais

- X e G arbitrários
  - Software de simulação pode ser muito complexo
    - Desempenho impraticável.
    - Mais utilizado para desenvolvimento de software.
- X e G idênticos
  - Muitas cópias da interface hardware-software de G em G.
  - Cada usuário com sua cópia privada da máquina G.
  - Escolha do SO para rodar em sua máquina privada.
  - Desenvolver/depurar seu próprio SO.
  - Simuladores não interferem um no outro.
  - *Slowdown* menor que para X diferente de G.

Máquinas virtuais

- Surgimento dos VMMs
  - Necessidade de simuladores mais eficientes de múltiplas cópias de uma máquina sobre seu próprio hardware.
- Parte do software para máquinas simuladas roda sobre o hardware, sem interpretação de software.

- Chamados de Virtual Machine Systems.
- Máquinas simuladas: máquinas virtuais (VMs).
- Software simulador: virtual machine monitor (VMM).

Máquinas virtuais

- VMM: interface única → ilusão de muitas máquinas.
- Cada interface (VM) é uma réplica eficiente do sistema de computação original.
  - Todas as instruções de processador (privilegiadas ou não).
  - Todos os recursos dos sistema (memória e E/S).
- VMs em paralelo → diversos SOs (núcleos privilegiados) concorrentemente.
- VMs fornecem réplicas isoladas de um ambiente em um sistema de computação.

Máquinas virtuais

- Recursos extras (CPU, memória) são usados pelo VMM.
  - Potencial queda na vazão do sistema.
- Manter estado do processador virtual.
  - Integridade de todos os registradores visíveis, bits de estado e locais de memória reservada (controle de interrupção) devem ser preservados.
  - Captura e simulação de instruções privilegiadas.
  - Suporte a paginação em máquinas virtuais.

Máquinas virtuais

- Podem ser utilizadas para manter sistemas antigos
  - Novos sistemas podem ser testados
  - Programas podem ser convertidos.
- Atualizar/adicionar novos dispositivos sem alterar SO da máquina virtual
  - VM já suporta o dispositivo virtualizado.
  - Usuário tira vantagem do dispositivo atualizado sem necessidade de alteração de software.

Máquinas virtuais

- Teste de softwares de rede.
- Confiabilidade de software através de isolamento.
  - VMM é provavelmente correta: pequena e verificável.
- Segurança de dados.

Máquinas virtuais

- VMM
  - Camada software/hardware físico programável,
  - Transparente ao software acima
  - Usa eficientemente o hardware abaixo dela.

- De forma similar:
  - Virtualização de rede
  - Virtualização de armazenamento
  - Fornecem capacidade de multiplexar, em um único recurso físico, vários sistemas virtuais isolados uns dos outros.

Máquinas virtuais de sistema

Máquinas virtuais de sistema

- VM
  - Fornece ambiente completo de sistema
- VMM
  - 1 plataforma de hardware.
  - Múltiplas VMs → múltiplos ambientes de sistema operacional independentes simultaneamente.
- Isolamento entre sistemas concorrentes no mesmo hardware.
  - Característica importante de máquinas virtuais.
  - Sem interferência em caso de falha.

Máquinas virtuais de sistema

- VMM fornece, primordialmente, replicação de plataforma.
- Problema central: dividir recursos de hardware limitados entre múltiplos sistemas operacionais convidados.
- VMM tem acesso e gerencia todos os recursos de hardware.

Máquinas virtuais de sistema

- SO convidado e suas aplicações são gerenciadas sob controle (escondido) do VMM.
- SO realiza instrução privilegiada ou acesso a recurso: VMM intercepta a operação, realiza verificações, e a realiza em nome do SO convidado.
- SO convidado não é ciente dessa camada.

Máquinas virtuais de sistema

- Para usuário, sistemas de VM fornecem essencialmente a mesma funcionalidade
  - Diferem na forma de implementação.
- Sistema de VMs clássico
  - VMM sobre o hardware.
  - Modo de privilégio mais alto.
  - Sistemas convidados: privilégios reduzidos
    - Permite a interceptação pela VMM.
    - Ações de SO convidado: seriam interação direta com o hardware; são tratadas pela VMM.

Máquinas virtuais de sistema

- Hosted VMs:
  - Software de virtualização roda sobre um sistema operacional hospedeiro.
  - Vantagem: usuário instala VMM como um software típico.
  - Software de virtualização pode se apoiar no sistema operacional hospedeiro para utilizar drivers de dispositivos e outros serviços de nível mais baixo.
  - VMWare GSX Server

Máquinas virtuais de sistema

- Paravirtualização:
  - Modificações no SO convidado.
  - Interface para um sistema similar mas não idêntico ao hardware nativo subjacente.
  - Interface de paravirtualização especialmente projetada para contornar características que tornam difícil a virtualização do ISA subjacente.
  - Ganho de desempenho; menor portabilidade e compatibilidade.
  - Ex.: Xen.

Máquinas virtuais de sistema

- Whole-system VMs:
  - Hospedeiro e sistema convidado podem não utilizar mesmo ISA (Ex. Windows / Power-PC).
  - Whole-system VMs virtualizam todo o software, incluindo SO e aplicações.
  - ISA diferentes: necessário emular códigos das aplicações e do SO.
  - Virtual PC.

Máquinas virtuais de sistema

- Multiprocessor virtualization
  - Hospedeiro é máquina grande e multiprocessada.
  - Particionar em sistemas menores multiprocessados
    - Distribui os recursos de hardware
  - Particionamento físico: recursos físicos separados para cada sistema virtualizado.
    - Alto grau de isolamento.
  - Particionamento lógico: hardware é multiplexado no tempo entre as diferentes partições.
    - Melhora utilização dos recursos.
    - Perde-se benefícios de isolamento de hardware.

Máquinas virtuais de sistema

- Codesigned VMs
  - Implementam ISA novo e proprietário focado em melhoria de desempenho e eficiência energética.
  - ISA do hospedeiro: novo ou extensão de ISA existente.
  - Não possui aplicações nativas.
  - VMM tem propósito de emular ISA do software convidado.
  - VMM reside em região de memória oculta dos softwares convencionais.
  - Inclui tradutor binário que converte instruções do convidado em sequências otimizadas de instruções do ISA do hospedeiro.

Máquinas virtuais de sistema

- Codesigned VM: Transmeta Crusoe
  - Hardware: VLIW.
  - Convidado: Intel IA-32
  - Economia de energia.

Clientes em SDs

Interfaces

- Grande parte de software de cliente é focada em interfaces (gráficas) para usuário.
  - Clientes magros
- Exemplo sistema X-Window
- Fig. 56

Transparência

- Partes da aplicação podem também ser executadas no lado cliente.
- Componentes para conseguir transparência de distribuição.
- Transparência de acesso: apêndice (stub) de cliente conforme definição da interface oferecida pelo servidor.
  - Fornece a mesma interface que está no servidor, mas oculta diferenças em arquiteturas de máquina e a comunicação.

Transparência

- Transparência de localização/migração: software do lado cliente pode saber localização.
  - Permite esconder localização geográfica do usuário.
- Transparência de replicação
  - Vincular-se a servidor de forma transparente.
  - Fig. 57
- Transparência de falha
  - Pode ser do lado do cliente.
  - Ex. tentar conexão várias vezes.

Servidores em SDs

Organização geral

- Processo que espera por requisições de serviço.
- Na prática, mapeamento um-para-um entre porta e serviço.
- Superservidores
  - Escutam várias portas (serviços independentes).
  - Iniciam processo do servidor quando chega requisição (inetd).
- Iterativo vs. concorrente:
  - Um vs. múltiplos clientes ao mesmo tempo.

Comunicação Fora da banda

- Comunicação fora da banda: permite enviar dados em canal separado.
  - Por ex. Interromper um servidor uma vez aceito (transferência de arquivo).
  - Porta de controle separada.

Servidores e estados

- Servidor sem estado: não mantém informações sobre estado dos clientes.
  - Servidor web.
  - Não lembra de clientes.
- Servidor com estado: mantém informações persistentes dos clientes.
  - Servidor de arquivos (FTP, dropbox).

Clusters de servidores

- Conjunto de máquinas conectadas por uma rede
  - Cada máquina executa um ou mais servidores
  - Conectados por rede local – alta largura de banda, baixa latência
- Organizado logicamente em 3 camadas
- Fig. 58.
- 1ª: comutador lógico: requisições de clientes roteadas
  - Variações.
  - TCP: aceitam requisições e repassam a servidor
  - Servidor web: aceita requisições HTTP, passa parte aos servidores de aplicação, processa e retorna requisição HTTP.

Clusters de servidores

- 2ª: Servidores de processamento/aplicação.
  - Servidores em hardware de alto desempenho
  - Servidores de baixa capacidade + armazenamento
- 3ª: servidores de processamento de dados/BDs
  - Máquinas especializadas para acesso rápido a disco
- Organização pode variar
  - Cada máquina pode ter seu disco local.
  - Por ex.: comum servidor de mídia em duas camadas.

Clusters de servidores

- Comutador pode precisar distinguir serviços
  - Encaminhar para servidor correto.
  - 1 máquina por app servidor evita interferência administrativa, tem hardware dedicado.
  - Ociosidade → máquinas virtuais.

- Importante: ocultar existência de diversos servidores
  - Aplicação cliente não sabem organização interna do cluster.
  - Transp. de acesso: comutador implementado em máquina dedicada.
  - Ponto de entrada com mesmo endereço de rede para todos os servidores.
  - Pode haver mais de um comutador.

Clusters de servidores

- Modo de acessar um cluster: estabelecer conexão TCP
  - Requisições de aplicação enviadas como parte de uma sessão.
  - Sessão termina com encerramento da conexão.
- Comutador de camada de transporte
  - Aceita requisições de conexão TCP
  - Transfere conexões a um dos servidores
  - Transferência TCP (TCP handoff)
  - Fig. 59.

Clusters de servidores

- Comutador recebe requisição TCP
- Identifica melhor servidor para manipulá-la
- Repassa o pacote de requisição para servidor
- Servidor envia reconhecimento de volta ao requisitante
- Ocorre spoof do endereço IP do comutador pelo servidor
  - Cliente espera resposta do comutador, não de outro IP
  - Modificações no nível do SO

- Importante na distribuição de carga
  - Ex: alternância cíclica (round robin)
  - Outros: distinguir carga da requisição e tipo de serviço.

Servidores distribuídos

- Clusters de servidores: estáticos
- Ponto de entrada é ponto de falha.
  - Oferecer diversos pontos de entrada
  - DNS pode retornar endereços diferentes para mesmo hospedeiro
  - Se um falha, cliente precisa refazer requisição DNS
- Outra opção: servidor distribuído
  - Conjunto de máquinas que muda dinamicamente
  - Vários pontos de acesso, possivelmente variáveis
  - Apresenta-se como máquina única ao mundo externo
  - Objetivo: servidor robusto, estável e de alto desempenho

Servidores distribuídos

- Como conseguir ponto de acesso estável?
- Usar serviços de redes
  - Suporte de mobilidade para IP versão 6 (MIPv6).

- MIPv6
  - Nó móvel tem uma rede nativa, onde normalmente reside e tem endereço estável (endereço nativo – h*ome address*)
  - Rede tem “repassador” conectado a ela (agente nativo), que toma conta do tráfego para o nó movel quando este estiver fora.

Servidores distribuídos

- Quando nó móvel se liga a uma rede externa, recebe endereço externo (Care-of Address – COA)
- Informado ao agente nativo
  - Repassa tráfego para o nó móvel
  - Comunicações externas usam endereço nativo, não COA
- Oferece endereço estável para servidor distribuído.
  - Endereço de contato designado ao cluster de servidores
  - Um nó servidor pode funcionar como ponto de acesso usando tal endereço, podendo ser assumido por outro nó
  - Registra seu COA no agente nativo

Servidores distribuídos

- Ponto de acesso pode cuidar da distribuição de requisições
- Se falha, outro ponto de acesso informa novo endereço COA
- Agente e ponto de acesso podem ser gargalos
  - Otimização de rota
  - Agente nativo pode passar endereço COA corrente para cliente, que pode comunicar-se diretamente.
  - Aplicação continuará usando endereço nativo, mas software básico subjacente traduzirá para o endereço COA

Servidores distribuídos

- Otimização de rota pode fazer clientes diferentes acreditarem estar usando um único servidor, mas cada um está comunicando com servidor diferente
- Fig. 60.

- C1 armazena par (HA, CA1)
- C2 armazena par (HA, CA2)
- Aplicações dos clientes tem ilusão que servidor é HA.

Gerenciamento de clusters de servidores

- Estender gerenciamento tradicional de uma máquina
  - Administrador pode conectar-se a cada nó
- Fornecer interface em máquina administradora
  - Coletar informações, atualizar componentes, adicionar/ remover nós, etc.
  - Facilita operações coletivas.
- Clusters muito grandes
  - Gerenciamento contínuo de falhas / atualizações.
  - $(1 – p)^n$ probabilidade de que todos estejam funcionando.
  - p=0,001, n=1000 → 36%

Gerenciamento de clusters de servidores

- Suporte a clusters grandes: ad-hoc
- Regras práticas existem, mas não há abordagem sistemática.
  - Soluções de autogerenciamento devem avançar.

Migração de código

Migração de código

- Passagem de programas, não apenas de dados.
- Tradicionalmente: migração de processo
  - Mover processo em execução.
  - Custoso, complicado.
  - Justificativa: desempenho pode melhora ao mover processos.

- Ex.: cliente-servidor, servidor é banco de dados.
  - BD muito grande, aplicação com muitas requisições
  - Pode ser melhor mover parte da aplicação para servidor.
  - Processamento próximo à origem dos dados.

Migração de código

- Também pode-se pensar em mover parte do servidor para cliente
  - Exemplo de pré-processamento de formulário
- Migração de código pode ajudar a melhor explorar paralelismo
  - Ex. agente móvel em buscas
  - Mesmo código despachado várias vezes
  - Aumento linear sem utilizar programação paralela

Migração de código

- Além de melhorar desempenho, também pode aumentar flexibilidade.
- Ao invés de quebrar aplicação e decidir previamente onde cada parte executa, pode ser configurado dinamicamente.
  - Ex: fornecer interface/código em tempo real, antes da invocação.
  - Fig. 61.
  - Requer protocolo para baixar código/inicializar.
  - Necessário que código possa ser executado na máquina cliente.

Migração de código

- Baixar software dinamicamente não requer todo software instalado com antecedência.
  - Tanto recebido quanto descartado quando necessário.
  - Por ex., pode-se alterar protocolos sem necessidade de atualizações manuais de software.
  - Problemas de segurança: garantir que código baixado é confiável (pode acessar outros dados...)

Migração de código - modelos

- Processo dividido em três segmentos
  - Segmento de código: conjunto de instruções que compõem o programa em execução
  - Segmento de recursos: referências a recursos externos de que o processo necessita (arquivos, dispositivos)
  - Segmento de execução: armazena estado da execução do processo (dados privados, pilha, contador de programa)

Migração de código - modelos

- Mínimo essencial: mobilidade fraca
- Possível transferir segmento de código (+ alguns dados de inicialização).
- Programa transferido é iniciado de acordo com posições pré-definidas.
  - Ex. applets java iniciam sempre do começo.
- Requer somente que máquina alvo possa executar aquele código (torná-lo portável)

Migração de código - modelos

- Outro modelo: mobilidade forte
- Segmento de execução também pode ser transferido.
- Processo em execução pode ser parado e movido
  - Retoma do ponto que parou
- Mais geral que mobilidade fraca, mas mais difícil de implementar.

Migração de código - modelos

- Outra distinção: quem inicia migração
  - Remetente
  - Destinatário

- Remetente: migração iniciada pela máquina em que o código está em execução.
  - Transferir programas para um servidor de computação
  - Enviar programa de busca, por exemplo.
- Destinatátio: iniciativa de migração tomada pela máquina-alvo.
  - Applets java.

Migração de código - modelos

- Destinatário:
  - Mais simples que pelo remetente.
  - Cliente toma iniciativa de migração.
- Contrário requer mecanismos de confiança entre cliente e servidor
  - Login/identificação
- Destinatário pode ser feita no anonimato
  - Servidor não se interessa pelos dados do cliente (?)
  - Migração serve para melhorar desempenho do cliente.

Migração de código - modelos

- Em mobilidade fraca: execução pelo processo-alvo ou novo processo.
  - Applets java são executados pelo browser.
  - Não há necessidade de novo processo.
  - Processo que executa precisa de alguma proteção em relação ao código a ser executado.

Migração de código - modelos

- Outro tipo de mobilidade forte: clonagem remota
  - Em contraste à migração de processos, produz cópia exata do processo original em máquina diferente.
  - Processo clonado: executa em paralelo com original.
  - Unix: bifurcação de processo e execução em máquina remota.

- Fig. 62.

Migração de código – segmento de recurso

- Segmento de recurso nem sempre pode ser transferido
  - Ex.: referência a uma porta TCP específica de uma conexão com outros processos.
  - Ao mover, precisa devolver a porta e requisitar uma nova.
- Mas pode não ser problema em outros casos
  - Referência a arquivo com URL absoluto.

Migração de código – segmento de recurso

- Três tipos de vinculação de processo-recurso
  - **Vinculação por identificador:** processo se refere a um recurso por seu identificador e requer exatamente o recurso referenciado.
  - Ex. Referências URL, terminais locais de comunicação.
  - **Vinculação por valor:** somente o valor do recurso é necessário.
  - Execução normal se outro recurso fornece mesmo valor.
  - Ex. Bibliotecas padronizadas (C, java...).

Migração de código – segmento de recurso

- **Vinculação por tipo**: processo precisa de um recurso com um tipo específico.
- Ex. dispositivos locais (monitor, impressora).

- Migração de código pode necessitar mudanças em referências.
- Se e como a referência deve ser mudada depende se recurso poder ser movido com código.
- Considerar vinculações recurso-máquina.

Migração de código – segmento de recurso

- Alterar vinculações recurso-máquina:
- **Recursos não ligados:** podem ser movidos com facilidade entre máquinas diferentes: arquivos de dados associados somente ao programa migrado.
- **Recurso amarrado:** pode ser possível mover ou copiar, mas a custos mais altos (bancos de dados locais, sites web completos).
- **Recurso fixo:** vinculados a uma máquina ou ambiente específico, não podem ser movidos (dispositivos locais).

Migração de código – segmento de recurso

- Nove combinações de vinculação: 3 processo-recurso e 3 recurso-máquina.
- Fig. 63.
- Por identificador:
  - **Não ligado:** mover recurso ou referência global, por ex. URL se compartilhado com outros processos.
  - **Amarado ou fixo:** criar referência global.

Migração de código – segmento de recurso

- Por valor:
  - **Não ligado:** copiar ou mover para novo destino. Se compartilhado, referência global.
  - **Amarrado:** bibliotecas: cópias geralmente disponíveis, caso contrário devem ser feitas antes da migração. Referência global quando muitos dados devem ser copiados.
  - **Fixo:** memória compartilhada entre processos. Referência global implicaria memória compartilhada distribuída.
- Por tipo:
  - Vincular novamente a recurso de mesmo tipo no local.
  - Se não estiver disponível, copiar ou mover, ou estabelecer referência global.

Migração de código – sistemas heterogêneos

- SDs: diferentes SOs, arquiteturas...
- Problemas similares aos da portabilidade
  - Assim como as soluções...
- Geração de código intermediário independente de máquina.
- Linguagens script e linguagens portáveis (java).
  - Máquina virtual ou intérprete de código fonte.
- Migração de ambientes inteiros (ex. migração de máquinas virtuais).

Migração de código – sistemas heterogêneos

- Migração de máquinas virtuais em tempo real.
  - Migrar toda a imagem da memória e vinculações a recursos locais.
- Migrar imagem da memória: 3 opções
  - Enviar as páginas de memória. Reenviar as que forme modificadas durante migração.
  - Parar máquina virtual, migrar memória e reiniciar na nova máquina.
  - Deixar a máquina “puxar” novas páginas conforme necessário (pós-cópia): copiar páginas sob demanda.

Migração de código – sistemas heterogêneos

- Parar máquina virtual: tempo ocioso pode ser inaceitável, dependendo do tipo de recurso fornecido pela máquina.
- Mover páginas sob demanda: pode prolongar eríodo de migração; pode resultar em mau desempenho (demora para conjunto de dados de um processo migrar).
- Cominar primeira e segunda opções: pré-cópia. Fase de parar para copiar é muito mais breve.
  - Tempos de parada de 200ms ou menos.
- Fig 64.