Clusters

Rísposli

Esse apresentação aborda o tema Clusters de maneira geral, procurando mostrar os conceitos básicos e os vários aspectos relacionados com a construção e utilização de sistemas para clusters.

Introdução

- Objetivo: Estabelecer a idéia de clusters, apresentar os conceitos básicos e, de forma geral, mostrar os diversos aspectos relacionados com o projeto de sistemas *clusters*.
- Apresentador: Luís Carlos Guimarães Rísposli
- Orientador: Prof. Dr. Sérgio Takeo Kofuji

Tópicos Abordados

- Estado Atual da Tecnologia
- Conceitos Básicos
- Classificação de *Clusters*
- Arquiteturas de *Clusters*
- Pontos importantes no projeto de *clusters*
- Suporte à Disponibilidade
- Suporte à Imagem Única
- Gerenciamento de *Jobs*
- Projetos de Pesquisa em *Clusters*
- Bibliografia

Estado Atual da Tecnologia

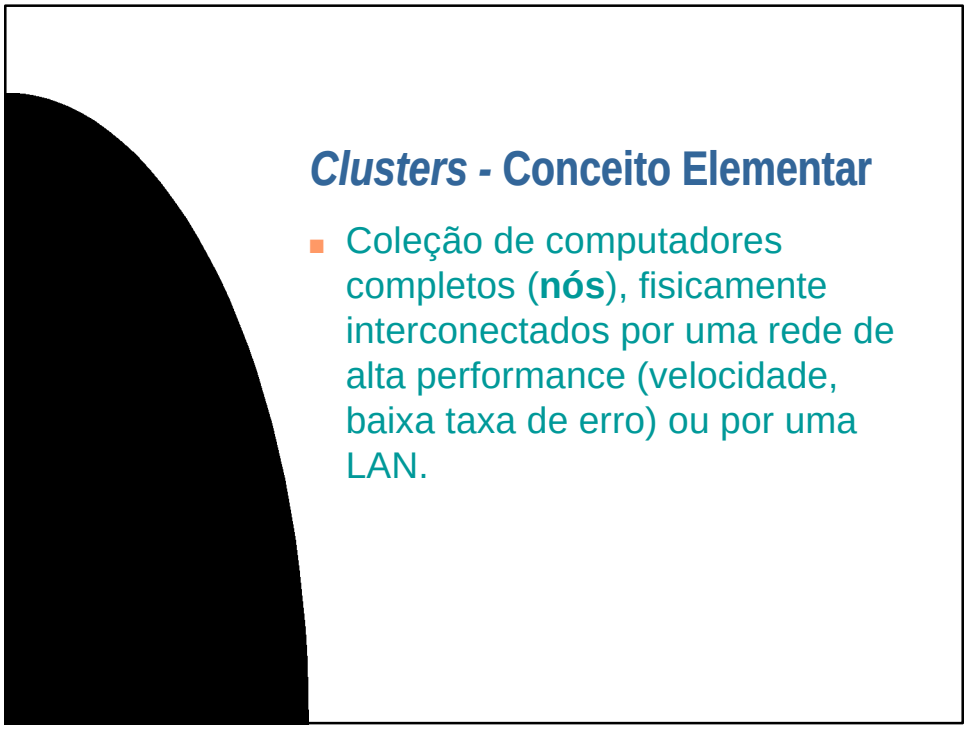
- Mais de 100.000 *clusters* de computadores estão em uso no mundo.
- A maioria dos *clusters* possuem tamanho na ordem de 10 **nós**.
- Muitos poucos *clusters* excedem 100 **nós**.

Estado Atual da Tecnologia

- O suporte a *clusters* está deslocando do mercado de grandes máquinas (*high end*) para o mercado de grande volume de máquinas pequenas (*high volume*).

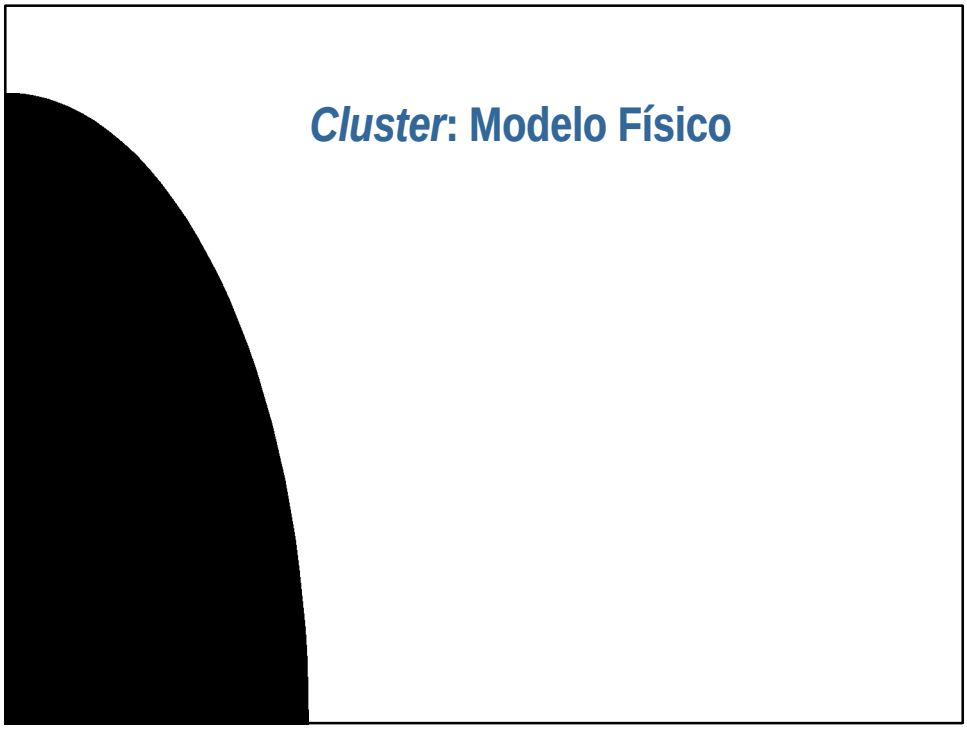
Conceitos Básicos

- *Cluster*
- Comparação de *clusters* com outras arquiteturas
- Benefícios e dificuldades envolvendo *clusters*


Clusters - Conceito Elementar
Coleção de computadores completos (**nós**), fisicamente interconectados por uma rede de alta performance (velocidade, baixa taxa de erro) ou por uma LAN.


Clusters - Conceito Refinado
Coleção de computadores completos (**nós**), fisicamente interconectados (por uma rede de alta performance ou LAN), onde todos os **nós** devem ser capazes de trabalhar coletivamente como um único e integrado sistema computacional, provendo serviços ininterruptos e eficientes.

Cluster: Modelo Físico



Clusters - Conceitos

- *cluster* envolve 5 importantes conceitos:
  - ✓ computador completo (**nós**);
  - ✓ rede de interconexão;
  - ✓ imagem única do sistema;
  - ✓ disponibilidade melhorada;
  - ✓ performance superior.


Computador Completo
Processador;
Memória;
adaptadores de I/O (disco etc);
sistema operacional completo para cada nó.

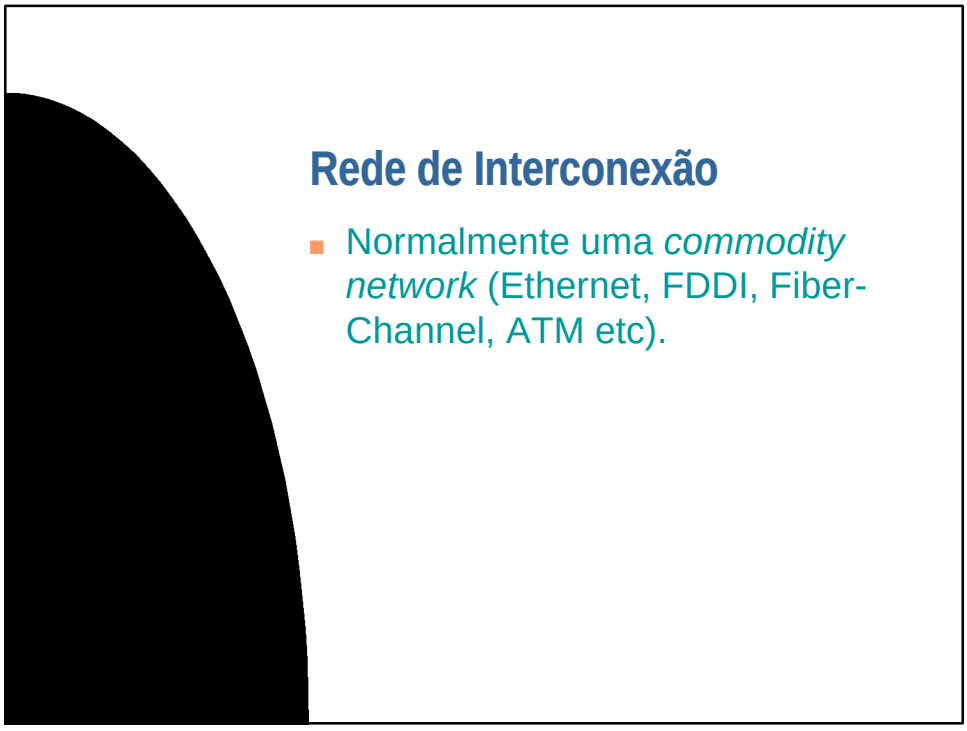

Rede de Interconexão
Normalmente uma commodity network (Ethernet, FDDI, Fiber-Channel, ATM etc).

Imagem Única

- essa característica de *cluster* contrasta com a de sistemas distribuídos (rede local de computadores) , onde os **nós** são usados como recursos individuais.

Melhor Performance

- Superservidor (cada **nó** serve m clientes, um cluster com n **nós** ⇒ m.n clientes);
- Processamento paralelo distribuído.

Cluster: Arquitetura Típica

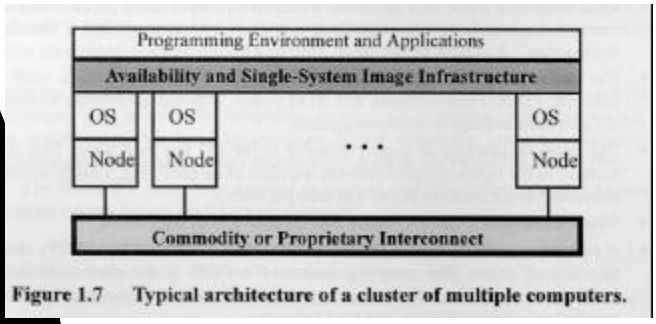
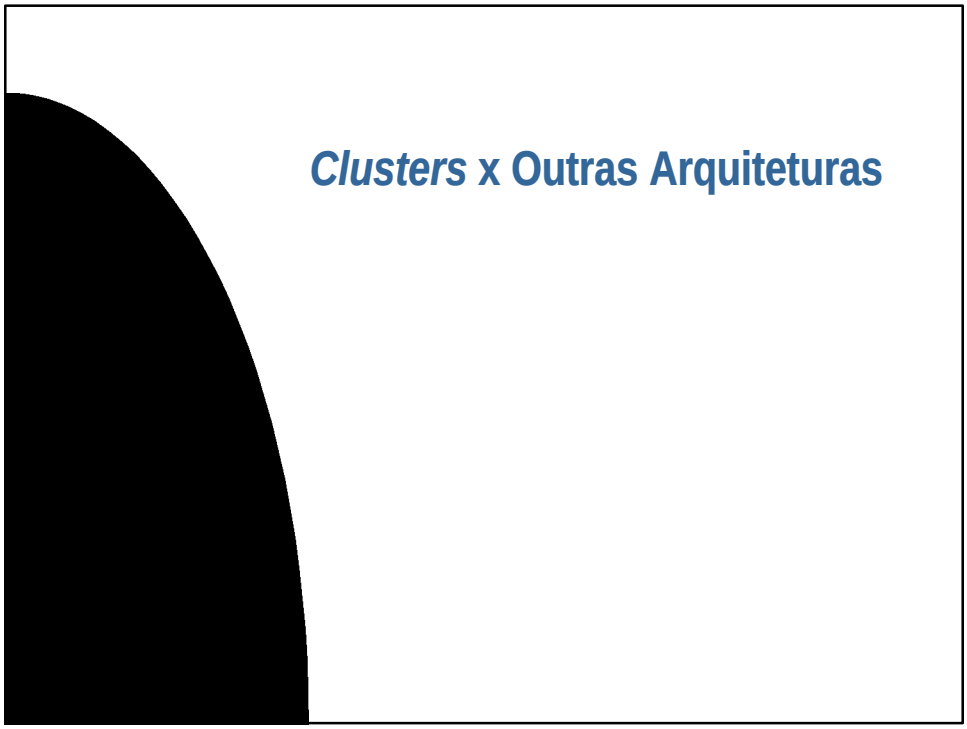
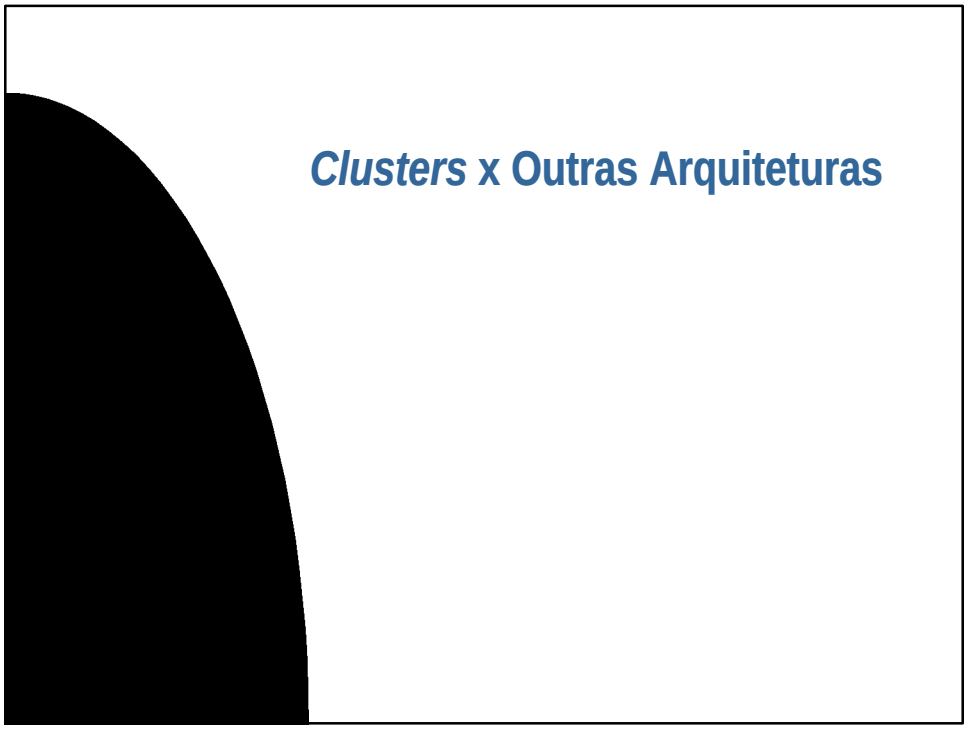


Figure 1.7 Typical architecture of a cluster of multiple computers.

Clusters x Outras Arquiteturas


Clusters x Outras Arquiteturas


Clusters x Outras Arquiteturas

Benefícios Potenciais de *Clusters*

- Facilidade de Uso (*Usability*);
- Disponibilidade (*Availability*);
- Escalabilidade (*Scalable Performance*);
- Boa relação performance/custo (*performance/cost ratio*).

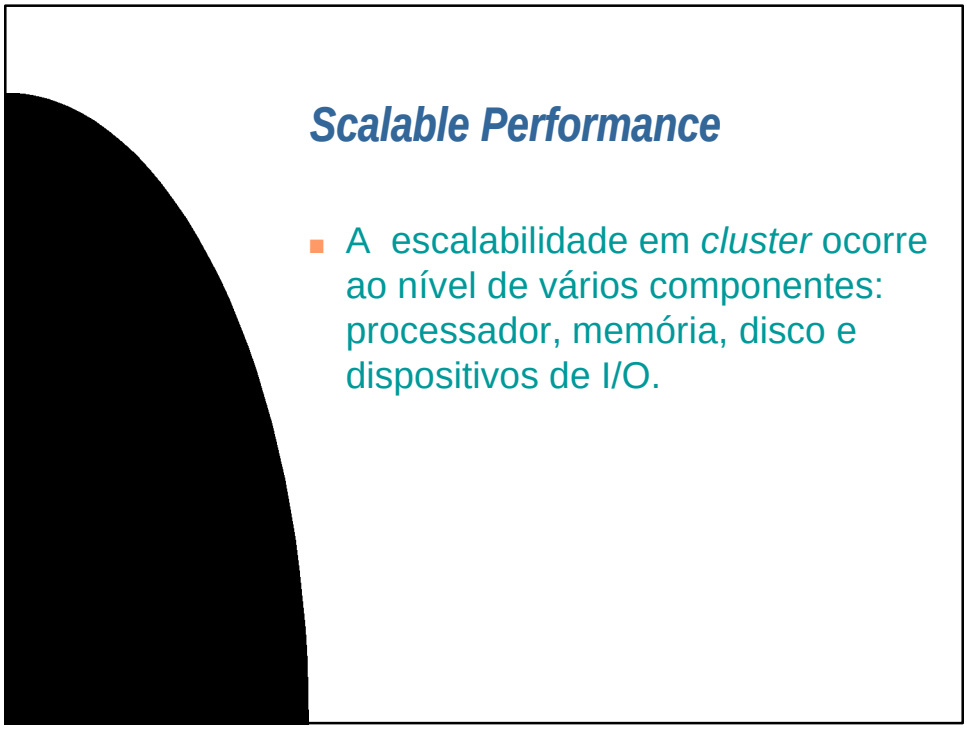
Usability

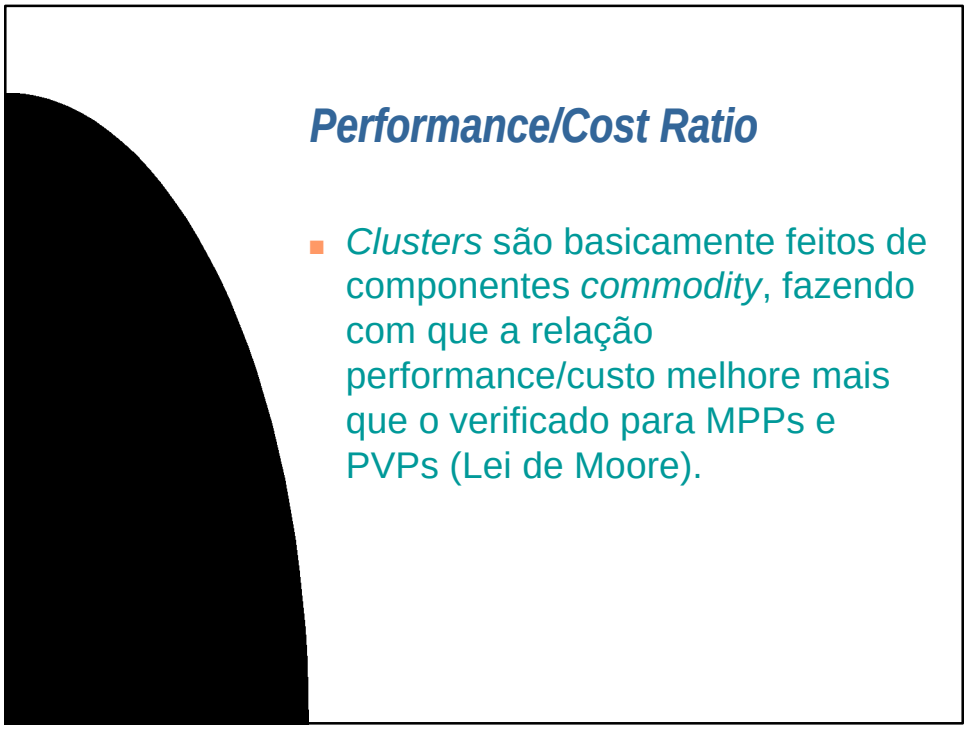
- Cada **nó** de um *cluster* deverá prover tudo que uma *workstation* oferece, suportando inclusive a execução de aplicações seqüenciais sem qualquer mudança.

Availability

- Potencialidade devida à redundância de componentes *commodity*: processadores e memória, discos locais, sistema operacional;
- Técnicas implementadas em *software* são utilizadas.

Availability Comparada


Scalable Performance
A escalabilidade em cluster ocorre ao nível de vários componentes: processador, memória, disco e dispositivos de I/O.


Performance/Cost Ratio
Clusters são basicamente feitos de componentes commodity, fazendo com que a relação performance/custo melhore mais que o verificado para MPPs e PVPs (Lei de Moore).

Lei de Moore (3 interpretações)

- O número de transistores num microchip duplica a cada 18-24 meses;
- A velocidade de um microprocessador duplica a cada 18-24 meses;
- O preço de um microchip cai aproximadamente 48% a cada 18-24 meses.

Performance Comparada *Cluster*

Classificação de *Clusters*

- Serão utilizados 4 atributos ortogonais (independentes):
  - Empacotamento;
  - Controle;
  - Homogeneidade;
  - Segurança.

Clusters segundo o Empacotamento

- Compacto: os **nós** estão próximos entre si, reunidos em 1 ou mais *racks*, e eles não estão conectados a periféricos - teclado, monitor, mouse etc. (*headless workstations*).
- Espalhado: os **nós** estão ligados aos periféricos comuns e podem localizar-se em salas distintas, prédios distintos e até mesmo estarem em locais distantes.

Clusters segundo o Controle

- Centralizado: todos os **nós** pertencem, são gerenciados e controlados por um operador central (administrador). *Clusters* compactos normalmente são centralizados.
- Descentralizado: cada **nó** poderá pertencer a um usuário distinto, que poderá reconfigurá-lo, atualizá-lo e efetuar *shut down*.

Clusters segundo a Homogeneidade

- Homogêneo: todos os **nós** usam a mesma plataforma (arquitetura do processador e SO).
- Heterogêneo: existem **nós** operando em plataformas diferentes.

Clusters segundo a Segurança

- Exposto: o meio de comunicação interno ao *cluster* fica disponível para acessos externos. Qualquer **nó** pode ser contactado através de um protocolo padrão (TCP/IP, por exemplo).
- Fechado (Protegido): o meio de comunicação interno ao *cluster* fica protegido de acessos externos.

Atributos para Classificação de *Clusters*

Clusters Dedicados (*Louis Turcotte*)

- instalados num *rack*, dentro de uma sala;
- tipicamente homogêneos;
- gerenciado pelo corpo de administradores;
- acesso feito através de um sistema *front-end*;
- usados como substitutos dos tradicionais *mainframes* ou supercomputadores.

Clusters Corporativos(*Louis Turcotte*)

- cada **nó** é um computador com todos os periféricos necessários;
- os **nós** estão espalhados, não necessariamente restringindo-se ao espaço de uma sala;
- os **nós** individualmente são possuídos por diversos usuários;
- os *jobs* locais possuem prioridade sobre os *jobs* corporativos;
- os **nós** normalmente são de natureza heterogênea e conectados através de rede de baixo custo (Ethernet).

Arquiteturas de *Clusters*

- Os **nós** de um *cluster* podem ser conectados de 3 formas diferentes:
  - Sem Compartilhamento (*Shared Nothing*);
  - Disco Compartilhado (*Shared Disk*);
  - Memória Compartilhada (*Shared Memory*).

Arquitetura *Shared Nothing*

- Os **nós** são interconectados via barramento de I/O.

Arquitetura *Shared Disk*

- Tipicamente utilizada para prover disponibilidade em clusters pequenos (*availability clusters*) que executam aplicações comerciais.

Arquitetura *Shared Memory*

Arquitetura *Shared Memory*

- É uma forma recente de construir *clusters.*
- Os nós se ligam à rede através do barramento de memória.
- Apresenta algumas dificuldades de implementação:
  - ausência de um padrão para barramento de memória aceito por todos;
  - barramento de memórias evolui mais rapidamente.

Projeto de Clusters: 4 Pontos Importantes

- Disponibilidade;
- Imagem única do sistema;
- Gerenciamento de *jobs*;
- Comunicação eficiente.

Disponibilidade

- *Clusters* naturalmente já são providos com bastante redundância: processadores, memórias, discos, dispositivos de I/O, múltiplas imagens do SO etc.
- Para que esse potencial reverta em disponibilidade é necessário o emprego de técnicas específicas.

Imagem Única do Sistema (SSI)

- Não basta conectar entre si várias *workstations* para obter um *cluster.*
- Não é uma meta fácil de se atingir.

Gerenciamento de Jobs

- *Clusters* devem usar de forma eficiente o conjunto de *workstations;*
- Gerenciamento de: execução em *batch,* execução interativa, balanceamento de carga, processamento paralelo etc.

Comunicação Eficiente

- Objetivo importante mas difícil de ser atendido, se comparado a MPPs:
  - os **nós** estão mais afastados, ficando mais expostos a problemas de comunicação.
  - uso de componentes *commodity* conjuntamente com protocolos padrão (TCP/IP).
  - inexistência de padrão para protocolos de baixo nível.

Arquitetura Ideal de um *Cluster*

Suporte à Disponibilidade

- Conceitos.
- Tipos de Falha.
- Técnicas de Melhoria;
- *Checkpointing* e Restabelecimento após Falha.

Conceitos

- 3 conceitos intimamente ligados (RAS):
  - Confiabilidade: medida do tempo em que o sistema opera sem parar;
  - Disponibilidade: percentagem do tempo em que o sistema fica disponível para o usuário;
  - *Serviceability*: caracteriza o grau de facilidade com que se presta serviço ao sistema (manutenção de *hardware*, *software*, reparos, atualizações etc)

Definição de Disponibilidade

- Disponibilidade = MTTF/(MTTF+MTTR)
- *Serviceability* → MTTR.
- Confiabilidade → MTTF.

“Alta” Disponibilidade???

Alguns Exemplos

Interrupção na Operação

- Falha Não Planejada: falha de *hardware*, desconexão da rede, falta de energia, falha do SO etc.
- Desligamento Planejado: o sistema é retirado de operação para manutenção, reconfiguração, atualização etc.

Tipos de Falhas

- Transiente: não é necessária a substituição de nenhum componente (*roll back*).
- Permanente: algum componente (*hardware/software*) deve ser reparado ou substituído.

Tipos de Falhas (cont.)

- Parcial: afeta somente uma parte do sistema, mas ele ainda continua operacional.
- Total: derruba todo o sistema, retirando-o de operação. Existência de *single point of failure*.

Exemplos: Pontos de Falhas

Técnicas de Melhoria da Disponibilidade

- Redundância Isolada;
- *Failover*;
- Restabelecimento.

Redundância Isolada

- elimina *single point of failure,* evitando falha total no sistema;
- possibilita efetuar reparo enquanto o resto do sistema continua operando;
- permite testes mútuos entre os componentes primário e secundário.

Exemplo

Configurações de Redundância

- *Hot Standby*: o componente secundário não faz nenhum trabalho, mas permanece pronto para entrar em ação.
- *Mutual Takeover*: não existe componente secundário; todos trabalham.
- *Fault-Tolerant*: redundância provida por N componentes; custo/benefício desfavorável.

Failover

- Na falha de um componente o restante do sistema assume o trabalho;
- Serviços: notificação de falha, diagnóstico e restabelecimento.
- Mecanismo usado: *heartbeat* (*heartbeat daemon*).

Esquemas de Restabelecimento

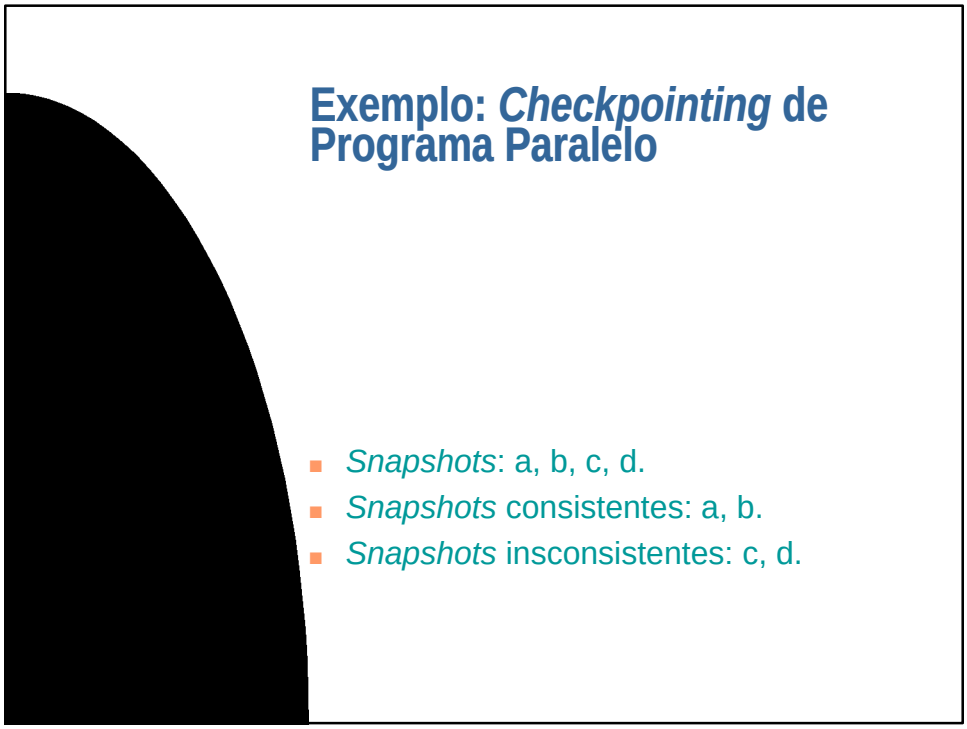
- Para trás (*backward*): o componente faltoso é isolado, o seu trabalho é assumido pelo sistema que volta a ser executado (*roll back*) partir de um estado anterior, previamente salvo;
- Para frente (*forward*): utiliza-se as informações de falha para reconstruir um estado válido do sistema e a partir daí continuar a execução. Dependente da aplicação.

Checkpointing

- Processo de salvar, periodicamente, num meio estável, o estado de um programa em execução, a partir do qual esta poderá ser reassumida após ocorrência de uma falha.

Intervalo de *Checkpointing*

- Intervalo ótimo = sqrt(MTTF. t_c/h)
- t_c: tempo gasto para salvar um *checkpoint*.
- $0 \leq h \leq 1$: percentagem de computação normal executada antes de uma falha, no intervalo de *checkpoint* .


Exemplo: *Checkpointing* de Programa Paralelo
Snapshots: a, b, c, d.
Snapshots consistentes: a, b.
Snapshots insconsistentes: c, d.


Checkpointing: Efeito Dominó
Checkpointing coordenado x *Checkpointing* independente.
Efeito dominó: resolvido com o registro de mensagens para *checkpointing* independente.

Suporte à Imagem Única (SSI)

- Carcaterísticas.
- Níveis Provedores de Imagem Única.
- Ponto de Entrada Único.
- Hierarquia de Arquivos Única.
- Ponto de Controle Único.
- Espaço de Memória Único.
- Ponto de Conexão de Rede Único.
- Outras Imagens Únicas Desejáveis.

Características

- Unicidade: o *cluster* por inteiro é visto pelo usuário como um sistema único.
- Controle Único: o administrador do sistema o configura através de um único ponto.
- Simetria: o usuário pode acessar os serviços do *cluster* a partir de qualquer **nó**.
- Transparência de localização: o usuário não tem ciência do componente do *cluster* que está provendo o serviço.

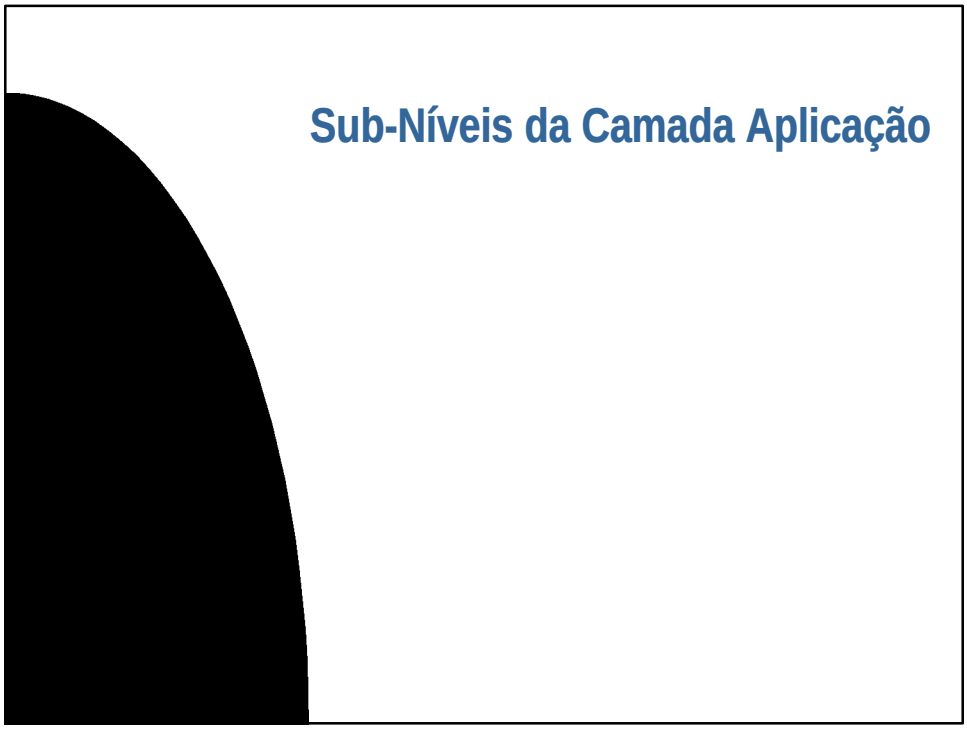
Níveis Provedores de Imagem Única

- A ilusão de um SSI pode ser obtida em diversos níveis de um sistema, e dentro de um mesmo nível pode-se ter imagens únicas distintas.
- Principais níveis:
  - Aplicação;
  - Acima do SO;
  - Sistema Operacional;
  - *Hardware*.

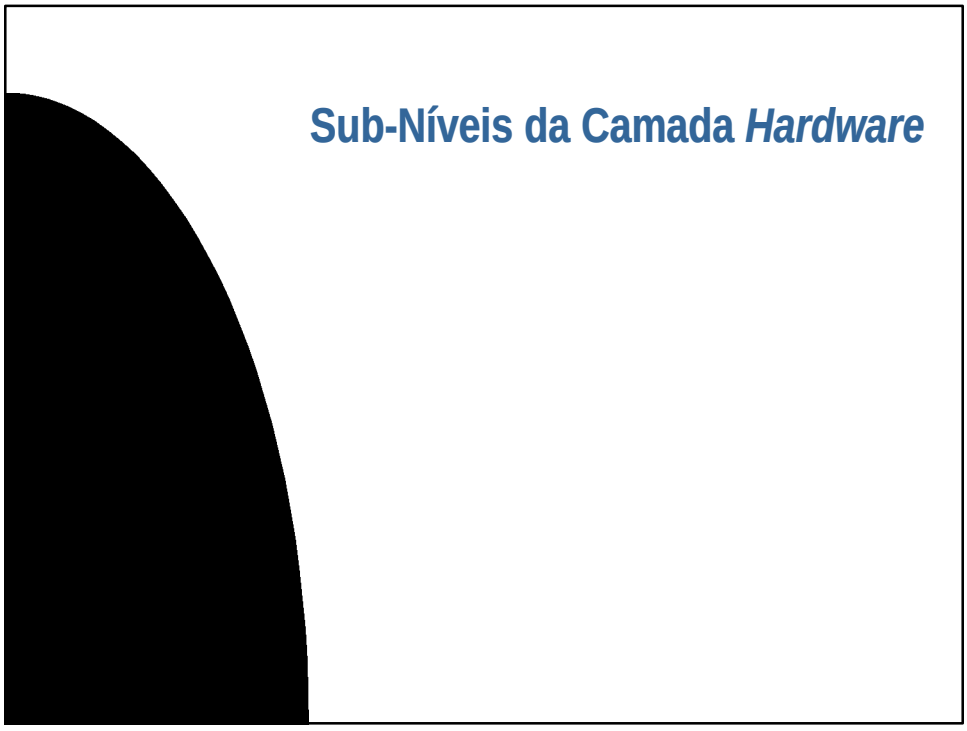
Exemplo de Níveis Provedores de Imagem Única

Exemplo de Imagens Únicas Distintas

Sub-Níveis da Camada Aplicação




Sub-Níveis da Camada Sistema Operacional


Sub-Níveis da Camada *Hardware*

Camada de Aplicação

- Exemplos: *browsers* para Web e BDs paralelos.
- Requer modificação de aplicações desenvolvidas para SMPs e *workstations*.

Camada do SO / *Hardware*

- Idealmente, SSI deveria ser provida pelo SO ou pelo *hardware*.
- Enormes dificuldades para criar SSI sobre *clusters* heterogêneos.
- Restringida a fabricantes (pois arquiteturas e SO's proprietários).

Camada Acima do SO

- Abordagem bastante promissora e viável pois é independente da plataforma e não requer modificações nas aplicações.

Ponto de Entrada Único

- Possibilita aos usuários conectarem-se (telnet, rlogin, http etc) ao *cluster* como se este fosse um único *host* virtual.

Exemplo

- Problemas:
  - Diretório *home.*
  - Autenticação.
  - Múltiplas sessões.
  - Falhas no *host.*

Hierarquia de Arquivos Única

- Cria ilusão de um grande sistema de arquivos que, transparentemente, integra todos os discos existentes no *cluster* (discos local e global).

Tipos de Repositórios numa Hierarquia de Arquivos Única

- Para qualquer processo executando num *cluster*:
  - Repositório Local;
  - Repositório Remoto;
  - Repositório Estável.

Repositório Estável

- Unidade logicamente centralizada.
- Seus diretórios formam a hierarquia de arquivos única.
- Fisicamente:
  - unidade centralizada (grande RAID) ou,
  - distribuído usando os discos locais dos **nós.**

Diretórios Visiveis a um Processo (UNIX)

- Diretórios do Sistema: /usr, /usr/local
- Diretório *Home*: ~/
- Diretório Global para Rascunho (Rep. Estável): /scratch ou /globalscratch
- Diretório Local para Rascunho (disco local): /localscratch.

Network File System (NFS)

- Arquivos em **nós** remotos são montados (ligados) numa hierarquia única.
- Usa RPC sobre TCP/IP.
- Pobre escalabilidade.

Andrew File System (AFS)

- Divide um sistema distribuído numa hierarquia de segmentos, cada um com seu servidor de arquivos.
- Usa protocolo com estado e *caching.*
- Boa escalabilidade mas apresenta lentidão.

Ponto de Controle Único

- Configurar, testar, monitorar e controlar todo o *cluster,* e cada nó individual através de um único ponto.
- Esse é um dos problemas mais desafiadores em *clusters*.
- ClusterView (HP).

Espaço de Memória Único

- Ilusão de grande memória centralizada, embora esteja fisicamente distribuída.
- Nativamente PVPs, SMPs e DSMs já tem essa característica.
- É também um empreendimento desafiador.

Ponto de Conexão de Rede Único

- Qualquer **nó** deverá ser capaz de ligar-se a qualquer ponto de conexão de rede (Fast Ethernet, ATM, FDDI, Ethernet).

Outras Imagens Únicas Desejáveis

- Um único espaço de I/O.
- Uma única interface com o usuário.
- Um espaço único para processos.
- Um único sistema de gerenciamento de *jobs*.

Gerenciamento de *Jobs*

- Sistema de Gerenciamento de *Jobs (JMS)*.
- Panorama de Sistemas de Gerenciamento de Jobs.

Sistema de Gerenciamento de *Jobs (JMS)*

- Servidor de Usuários (*User Server*): permite ao usuário submeter *jobs* a uma ou mais filas, apagar jobs, consultar estado de um *job*/fila etc.
- Escalonador de *Jobs* (*Job Scheduler*): escalona *jobs*/filas para serem executados, considerando recursos disponíveis, tipos de *jobs* etc.
- Gerenciador de Recursos (*Resource Manager*): faz alocação e reserva de recursos.

Características do JMS

- Funcionalidades realizadas de forma distribuída.
- Capacidade para reconfigurar dinamicamente o *cluster* com mínimo de impacto sobre os *jobs* que estão executando.
- O usuário deverá ser capaz de matar seus *jobs* de forma segura.
- O administrador do sistema deverá ser capaz de suspender/matar qualquer *job*.

Job Local x *Job* de *Cluster*

- *Job* Local: são criados pelo usuário dono do **nó**, fora do JMS.
- *Job* de *Cluster*: é criado e submetido através do JMS.

Jobs de *Cluster*

- *Job* serial: executa num único **nó**.
- *Job* Paralelo: executa em múltiplos **nós**.
- *Job* Interativo: os seus resultados são mostrados num terminal; requer resposta rápida.
- *Job Batch*: normalmente demanda por mais recursos; não requer resposta rápida.

Características de *Workloads* de *Clusters*

- Aproximadamente 50% dos *jobs* paralelos são submetidos durante horas regulares de trabalho.
- Quase 80% dos *jobs* paralelos gastam no máximo 3 minutos.
- *Jobs* paralelos que gastam no mínimo 90 min são responsáveis por 50% do tempo total de execução.

Características de *Workloads* de *Clusters*

- 60% a 70% das *workstations* ficam disponíveis para *jobs* paralelos, mesmo durante as horas de pico de processamento seqüencial.
- Aplica-se a regra 2:1: uma rede de 64 *workstations* pode sustentar uma *workload* paralela sobre 32 **nós** além da carga seqüencial original da rede ⇒ *clustering* dá, de graça , um supercomputador com a metade de **nós**.

Escalonamento de *Jobs*

- Prioridades:
  - Estática: esquema fixo.
  - Dinâmica:a prioridade do *job* pode alterar com o transcorrer do tempo (dia e noite).
- Alocação de recursos:
  - Estática: reserva fixa de recursos.
  - Dinâmica: recursos podem ser obtidos/liberados durante execução.

Modelos de Utilização dos Nós

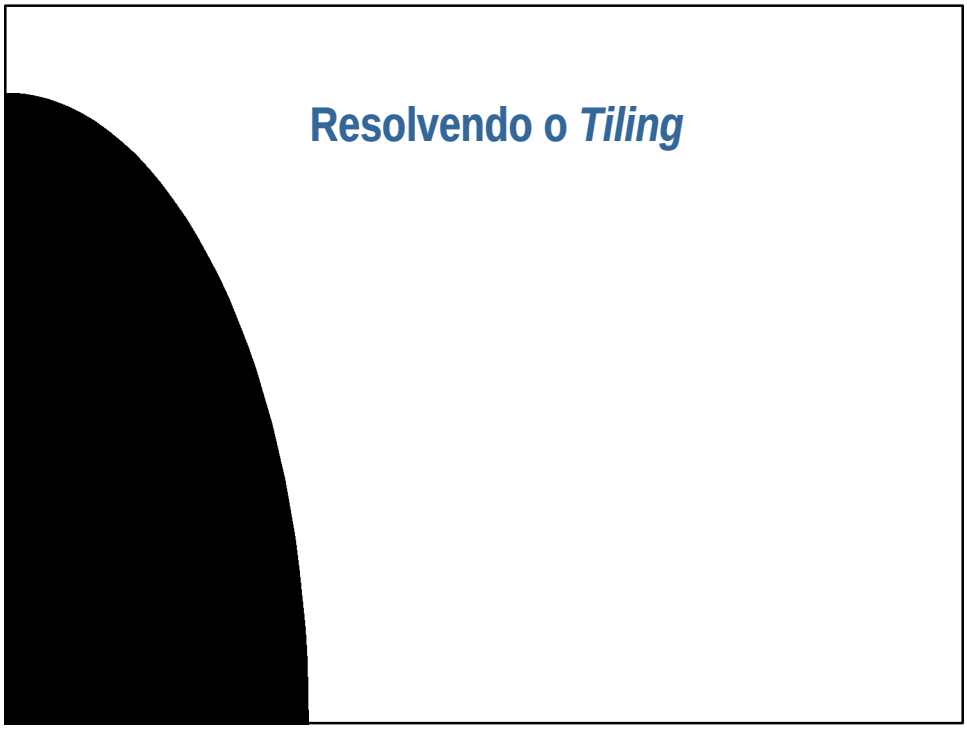
- Modo Dedicado.
- *Space Sharing.*
- *Time Sharing.*

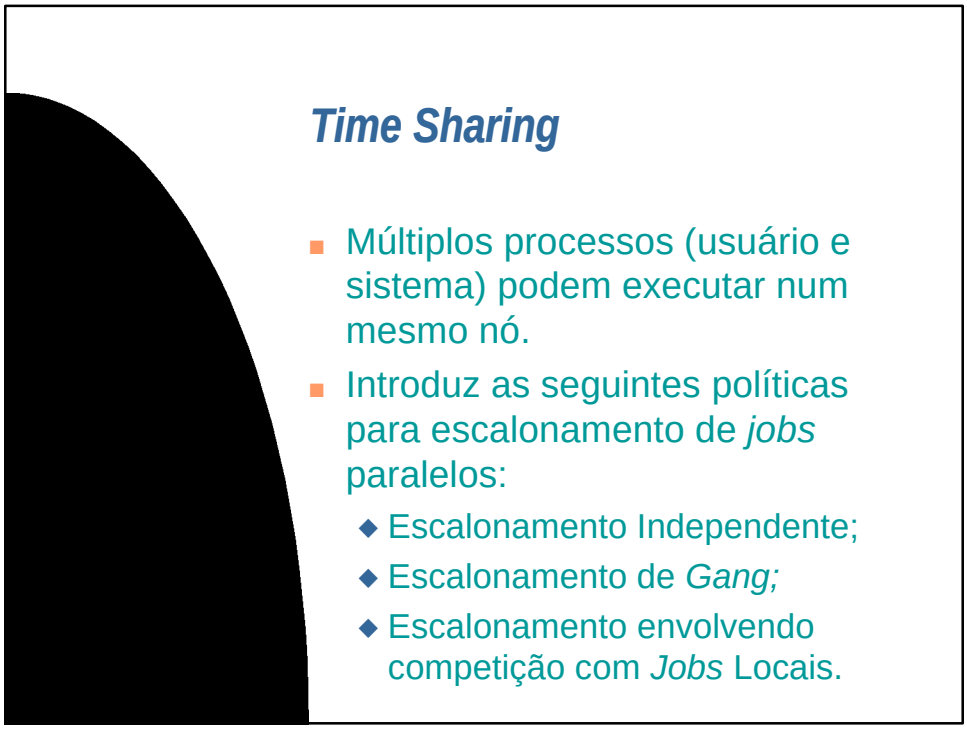
Modo Dedicado

- Somente um *job* é executado no *cluster,* num dado tempo, e no máximo um processo do *job* é atribuído a cada **nó**.
- Má utilização do sistema.

Space Sharing

- Múltiplos *jobs* podem executar simultaneamente, mas sobre partições disjuntas do *cluster;* também atribui-se um processo do *job* a cada **nó**.
- 2 problemas: *Tiling* e *jobs* grandes.


Resolvendo o *Tiling*


Time Sharing
Múltiplos processos (usuário e sistema) podem executar num mesmo nó.
Introduz as seguintes políticas para escalonamento de *jobs* paralelos:
Escalonamento Independente;
Escalonamento de *Gang;*
Escalonamento envolvendo competição com *Jobs* Locais.

Escalonamento Independente

- utiliza o próprio SO do **nó** para fazer o escalonamento de diferentes processos.
- Problema: *jobs* paralelos interagem entre si.

Gang Scheduling

- Todos os processos de um *job* paralelo são escalonados juntos.
- Pode melhorar a performance por até duas ordens de magnitude.
- *Gang-scheduling skew*: máxima diferença entre o tempo de início da execução do primeiro e último processos do *job* paralelo.

Competição com *Jobs* Locais

- *Jobs* locais devem possuir prioridade sobre *jobs* de *cluster*.
- Duas abordagens:
  - Migração: o *job* de *cluster* procura outro **nó** com menor carga;
  - Permanência: o processo de *cluster* executa com a menor prioridade (processos do *kernel*, processos locais e processos de *cluster*).

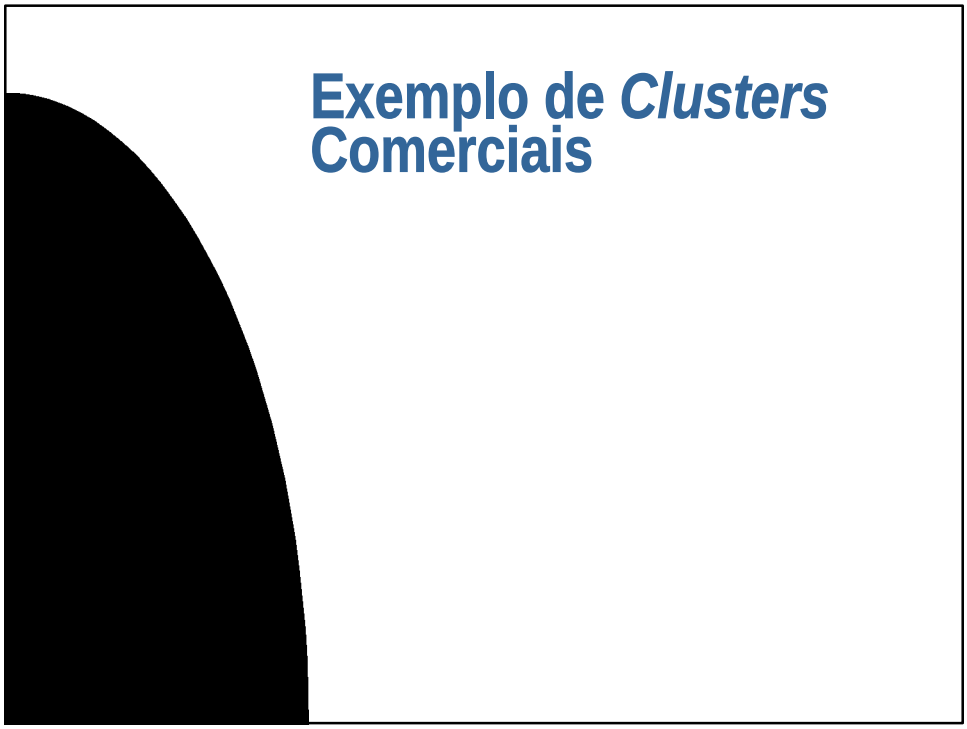
Esquemas de Migração

- Disponibilidade de **Nós**: estudos mostram que, mesmo durante horas de pico, 60% das *workstations* de um *cluster* estão livres.
- *Overhead* de Migração: estudos apontam para uma queda de 2.4 na velocidade de processamento.
- Threshold de Recrutamento: indica a quantidade de tempo que uma *workstation* deve permanecer sem uso antes do sistema considerá-la ociosa. (estudo: 3 min).

Síntese

Panorama de JMSs

Exemplo de *Clusters* Comerciais



Projetos de Pesquisa em *Clusters*