# Tolerância a Falhas

Rodrigo D. Malara
Adaptado de Guilherme Bertoni Machado

**Universidade de Araraquara**
Engenharia de Computação
Sistemas de Informação

# Ariane 5

- Lançado em 4/6/1996 pela European Space Agency
- 10 anos de projeto, ao custo de 7 bilhões de dólares
- Foguete e sua carga avaliados em 500 milhões de dólares
- Explodiu 40 segundos depois do lançamento
- Motivo: conversão de um número de ponto flutuante de 64 bits
- (correspondente à velocidade horizontal do foguete)
- para um valor inteiro de 16 bits (ao reusar um módulo da Ariane 4)

# Therac-25

- Máquinas de aceleração linear utilizadas no tratamento de câncer
- East Texas Cancer Center, Tyler, entre 1985 e 1987
- Quatro pessoas em tratamento médico de câncer receberam doses letais de radiação
- Motivo: falha de programação na coordenação de tarefas concorrentes (condição de disputa).

# Míssel Americano Patriot

- Guerra do Golfo, fevereiro de 1991
- O Patriot falhou em interceptar um míssel Scud (lançado pelo inimigo), o qual matou 28 soldados americanos em suas barracas, em Dahran, Arábia Saudita.
- Motivo: erro no cálculo do tempo devido a um arredondamento na representação do valor 1/10 usando um registrador de 24 bits. (O erro final chegou a 34 décimos de segundo, tempo suficiente para o Scud percorrer quase meio quilômetro).

# Airbus A320

- Varsóvia, setembro de 1993
- A aeronave, ao pousar, bateu num morro e duas pessoas morreram.
- Pilotos esperavam ter vento *contra* a aeronave no momento do pouso, por isso aumentaram a velocidade da aeronave.
- Mas tiveram vento *a favor*, fazendo o avião ganhar muita velocidade no pouso.
- Por isso, os sensores nas rodas não sinalizaram que a aeronave havia pousado, impedindo que os freios fossem acionados.

# Dragonair's A320

- Hong-Kong, junho de 1994
- Pouso complicado
- Um forte vento impediu o movimento dos *flaps* no momento do pouso.
- Mas, o sistema acreditou que os flaps estavam na posição correta, causando sérios problemas na operação de pouso.

# Boeing B757

- Colômbia, dezembro de 1995
- Aeronave bateu numa montanha, matando 160 pessoas
- Motivo: sistema de gerenciamento de vôo estava operando com uma base de dados de navegação incorreta, fazendo com que a tripulação recebesse informação errada sobre a posição da aeronave.

# Intel Pentium Processor

- 1995
- Algoritmo de divisão utiliza uma tabela de busca com 1066 entradas.
- Somente 1061 entradas foram carregadas na seção PLA devido a um erro num *loop.*
- Gravado em silício e nunca verificado.
- Efeito: algumas divisões com ponto flutuante geram resultados errados na quarta casa decimal.
- Custo da substituição: mais de 400 milhões de dólares.

# Polícia France

- Paris, 1989
- 41.000 motoristas com infração de trânsito foram notificadas que haviam cometido crimes como assassinato, tráfico de drogas, extorsão e prostituição, ao invés de suas violações de trânsito.

# Internet Worm

- Novembro de 1988
- 6.000 computadores na Internet caíram em poucas horas.
- Semanas se passaram até que voltassem a operar normalmente.
- Motivo: um hacker (estudante de graduação) criou um sistema que continha um erro de programação e um erro matemático em seu código, fazendo com que se replicasse muito rapidamente.

# Dependabilidade

- Availability (disponibilidade)
- Reliability (confiabilidade)
- Safety (segurança no funcionamento)
- Maintainability (capacidade de manutenção)

# Disponibilidade

- Propriedade de um sistema estar pronto para ser usado imediatamente.
- Probabilidade de o sistema estar funcionando corretamente em qualquer momento determinado.
- Um sistema de alta disponibilidade tem alta probabilidade de estar funcionando num dado momento.

# Confiabilidade

- Propriedade de um sistema poder funcionar continuamente sem falha.
- Definida em termos de um intervalo de tempo.
- Um sistema de alta confiabilidade provavelmente continuará a funcionar sem interrupção durante um período de tempo relativamente longo.

# Disponibilidade *versus* Confiabilidade

- Um computador que fica fora do ar por um milissegundo a cada hora tem alta disponibilidade (99,9999%), mas baixa confiabilidade.
- Um computador que nunca falha mas é sempre desligado duas semanas por ano tem alta confiabilidade, mas somente 96% de disponibilidade.

# Segurança no funcionamento

- Se um sistema deixar de funcionar corretamente durante um certo tempo, nada de catastrófico acontecerá.
- Exemplo: sistemas de controle de usinas de energia nuclear. Sistema potencialmente inseguro.

# Capacidade de manutenção

Facilidade com que um sistema que falhou pode ser consertado.

- Modularização
- Documentação

Um sistema com grande capacidade de manutenção tende a ter alta disponibilidade.

# Conceitos

Estados de um Sistema:

Sistema próprio: serviço é fornecido pelo sistema como foi especificado (ou nominal em inglês)
Sistema interrompido ou impróprio: serviço não é fornecido conforme especificado

# Conceitos

Confiabilidade pode ser representada por:

- Tempo Médio Para a Falha (MTTF): indica o tempo médio que o sistema fica sem falhar
  - Ex.: X horas ou dias de funcionamento
    Quanto maior, melhor
- Tempo Médio Entre Falhas (MTBF): representa o tempo entre falhas sucessivas
  - Ex.: Y horas ou dias entre falhas
    Quanto maior, melhor
- Probabilidade (taxa) de Falha
  Ex.: $10^{-Z}$ falhas/hora ou falhas/dia
  Quanto maior o expoente, melhor

# Conceitos

Manutenibilidade é representada por:

- Tempo Médio Para Reparo (MTTR): indica o tempo necessário para que o sistema volte a funcionar corretamente
  - Ex.: X segundos, minutos ou horas para o sistema voltar a funcionar corretamente
    Quanto menor, melhor

    Para hardware, geralmente em contrato e é expresso através de uma SLA
    - Service Level Agreement ou Acordo de Nível de Serviço

    Quantidade de réplicas ou equipamentos sobressalentes

# Conceitos

- Disponibilidade é representada por:
  - Grau de disponibilidade: MTBF/(MTBF + MTTR)
    - Ex.: sistema disponível por 99.9% do tempo
    - Quanto maior, melhor

| Disponibilidade | Tempo de parada / ano |
|---|---|
| 90% | ~ 1 mês |
| 99% | ~ 4 dias |
| 99.9% | ~ 9 horas |
| 99.99% | ~ 1 hora |
| 99.999% | ~ 5 minutos |
| 99.9999% | ~ 30 segundos |

# Conceitos

Segurança no funcionamento (seguridade/safety) é representada por:

- Grau de Seguridade: probabilidade do sistema ser recuperável (de não se tornar impróprio) em caso de falha, ou seja, a chance de uma falha não ser catastrófica
- Falhas  Benignas/(Falhas Benignas + Catastróficas)
  - Ex.: sistema recuperável em 98% das falhas
    Quanto maior, melhor

# Conceitos

## Falta, Erro e Falha

- Faltas que ocorrem em sistemas
  - Originadas por fatores internos ou externos
    Podem ficar dormentes até serem notadas
  - Algumas faltas também são conhecidas como BUGS
  - Ou simplesmente componentes inoperantes
- Erros são resultantes de faltas no sistema
  - Ocorrem quando faltas impedem o funcionamento normal do sistema
    Diferentes faltas podem causar o mesmo erro
- Falhas podem ocorrer devido a erros no sistema
  - A falha é o efeito observável do erro
  - Diversos erros podem levar à mesma falha

# Conceitos

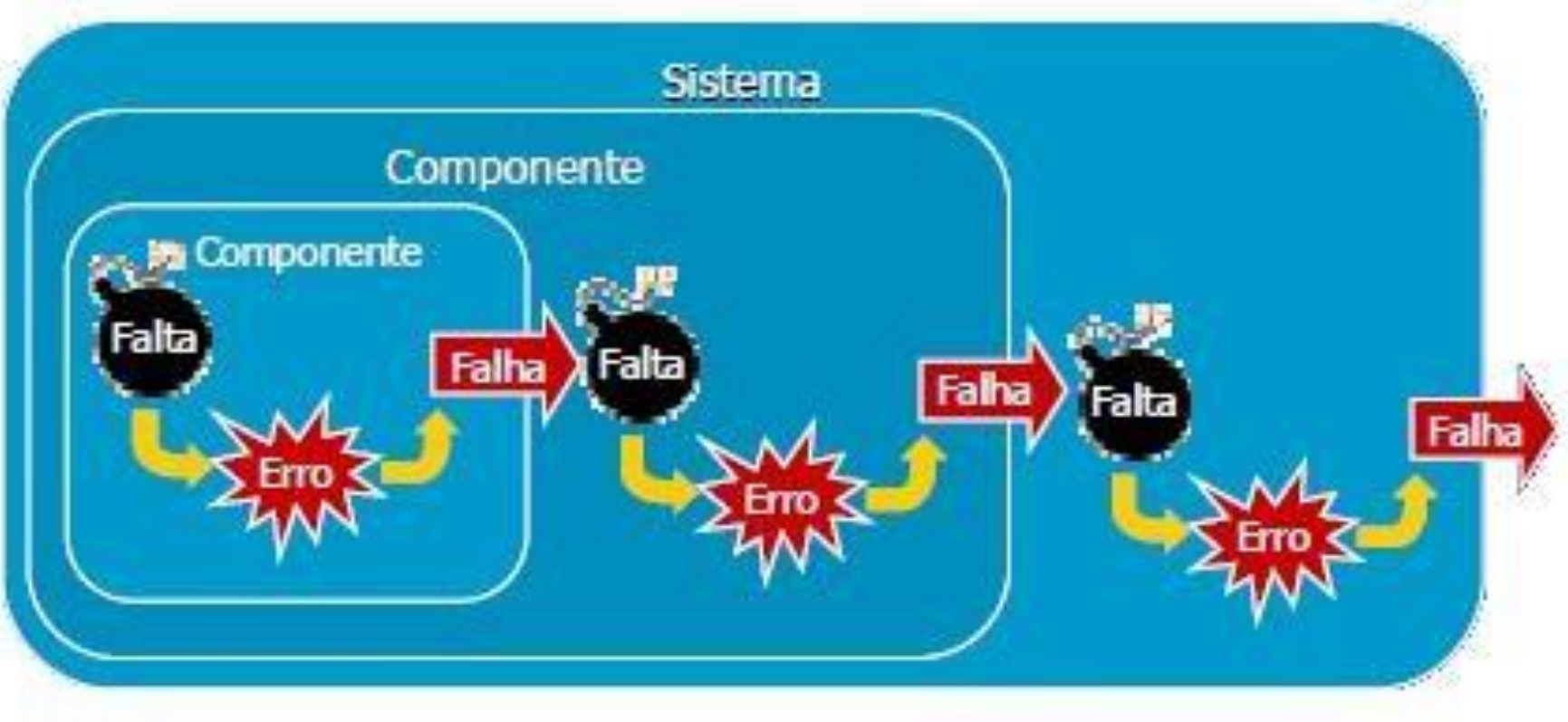
## Falta, Erro e Falha

### Exemplo: HD

- Um setor do disco pode estar com defeito (falta)
- Um erro de leitura pode ocorrer se um programa tentar ler ou escrever neste setor
- Pode ocorrer uma falha em um sistema que tente acessar este setor e não consiga
- Se os dados gravados neste setor estiverem replicados em outro local, o sistema pode tolerar a falta e não apresentar falha

# Conceitos

## Falhas em Cascata



# Conceitos

## Falhas em Cascata

A falha de um componente pode ocasionar a falha de outro que necessita dos serviços do primeiro, e assim sucessivamente, podendo levar o sistema como um todo a falhar

Exemplo:

- Uma falta no disco pode causar uma falha no sistema de arquivos
- Os servidores Web e de e-mail, que usam o sist. de arquivos, podem falhar
- Uma aplicação de comércio eletrônico baseada na Web pode também falhar

# Conceitos

## Previsão de Faltas

Estima a probabilidade de que defeitos ocorram  Permite que se avalie os riscos de falha

## Remoção de Faltas

Consiste em detectar e remover as defeitos antes que  causem erros e falhas
Usar ferramentas como debugger, scandisk, ...

## Prevenção de Faltas

Elimina as condições que fazem com que faltas ocorram durante a operação do sistema
Usa replicação interna, técnicas de validação, ...

# Conceitos

## Tolerância a Faltas

# Conceitos

## Tolerância a Faltas

Propriedade de sistemas que não falham necessariamente ao se deparar com uma falta

## Sistemas Tolerantes a Faltas

São sistemas capazes de tolerar faltas encontradas durante a sua execução

## Técnicas de Tolerância a Faltas

Permitem prevenir falhas contornando as faltas que os sistemas podem vir a apresentar

# Tipos de Faltas

Classificação em relação à sua origem:

- Física: causada pelo hardware
- De projeto: introduzida durante a fase de projeto do sistema
- De interação: ocorrida nas interfaces entre componentes do sistema ou na interação com o mundo exterior

Classificação em relação à sua natureza:

- Acidental ou Intencional
- Maliciosa ou Não

# Tipos de Faltas

Classificação em relação ao seu surgimento:

- Na fase de desenvolvimento do sistema
- Na fase de operação do sistema

Classificação em relação à sua localização:

- Interna ou Externa

Classificação em relação à persistência:

- Temporária
- Transiente ou Intermitente
- Permanente

# Tipos de Faltas

Classificação com base no modelo de faltas:

- Faltas Omissivas
  - Crash: deixa de funcionar permanentemente
  - Omissão: sistema deixou de fazer o que deveria em um determinado instante
  - Temporal: sistema atrasou-se para executar uma determinada ação
- Faltas Assertivas
  - Sintática: formato da saída é inadequado
  - Semântica: saída apresenta valor incorreto
- Faltas Arbitrárias: omissivas + assertivas

# Tipos de Faltas

Os tipos de faltas mais freqüentes são:

Faltas de operação e administração: 42%
Faltas de software: 25%
Faltas de hardware: 18%
Faltas de ambiente: 14%
Fonte: Jim Gray. Why do Computers Stop and What Can Be Done About It? IEEE SRDS'85.
Estudos mais recentes confirmam estes dados

# Replicação

Tolerância a faltas pode ser obtida através do uso de recursos redundantes

Redundância pode ser aplicada das seguintes maneiras:

- Redundância Temporal: repetir uma mesma tarefa até que um resultado válido seja obtido
- Redundância de Valores: replicar um dado armazenado ou enviado pela rede
- Redundância Espacial: usar várias réplicas de um componente de hardware ou software

# Replicação

Uso de réplicas aumenta a disponibilidade

Exemplo: se a probabilidade de perda de uma mensagem na rede é de 2% (disponibilidade de 0,98), se duplicarmos todas as mensagens, a chance de se perder as duas cópias será de 0,04% (disponibilidade de $1\text{-}0{,}02^2=0{,}9996$)

Exemplo 2: se um servidor fica indisponível durante 8 horas a cada ano (disponibilidade de 0,999), se criarmos 3 réplicas teremos uma parada total de 3,15 segundos em um século (disponibilidade de $1\text{-}0{,}001^3= 0{,}999999999$)

# Replicação

O acesso a serviços ou dados replicados deve ser transparente para o usuário

- Usuário deve acessar o dado ou serviço replicado da mesma forma que o faria se não houvesse replicação
- Se for preciso manter a consistência dos dados replicados, este processo deve ser efetuado automaticamente pelas réplicas
- Obs: Mesmo que mais de uma réplica responda a uma requisição, apenas uma resposta deve ser entregue ao usuário

# Replicação

## Técnicas de Replicação

Definem como as réplicas se comportarão durante o funcionamento normal do sistema e sob a presença de faltas

Principais Técnicas de Replicação

- Replicação Passiva (Primário-Backup)
- Replicação Ativa
- Replicação Semi-Ativa (Líder-Seguidores)
- Replicação Preguiçosa (Lazy)

# Replicação

## Replicação Passiva (Primário-Backup)

São criados um ou mais backups de um componente (primário) com o objetivo de substituí-lo em caso de falha
Funcionamento com propagação de estado instantânea:

Primário recebe requisições, as executa, atualiza o estado dos backups e retorna o resultado ao cliente

# Replicação

## Replicação Passiva (Primário-Backup)

# Replicação

## Replicação Passiva (Primário-Backup)

Em caso de falha do primário, um backup será escolhido para assumir o seu lugar

O backup escolhido terá o mesmo estado do primário até a última requisição executada

Uma requisição em execução durante a falha pode ser recuperada pelo cliente reenviando a requisição ao novo primário

Atualização de estado a cada requisição causa uma sobrecarga considerável no primário

# Replicação

## Replicação Passiva (Primário-Backup)

Funcionamento com *log* e *checkpoints*:

As requisições de clientes são enviadas ao primário e ao(s) backup(s)

Primário executa as requisições e responde aos clientes

Backup recebe as requisições mas não as executa - apenas as registra em um *log*

O estado do primário é consolidado no(s) backup(s) em instantes predeterminados - chamados de *checkpoints*

# Replicação

## Replicação Passiva (Primário-Backup)

Funcionamento com *log* e *checkpoints* (cont.):
O backup limpa o *log* a cada *checkpoint*

Em caso de falha do primário, o backup escolhido para assumir o seu lugar terá o estado do primário no último *checkpoint*

Para chegar ao mesmo estado do primário no instante da falha, o backup escolhido executa as requisições registradas no *log*

# Replicação

## Replicação Passiva (Primário-Backup)

# Replicação

## Replicação Passiva (Primário-Backup)

Procedimento executado a cada Checkpoint :

# Replicação

## Replicação Passiva (Primário-Backup)

### Considerações:

- O(s) backup(s) consomem muito pouco poder de processamento, pois não precisam processar as requisições
- O primário tem a obrigação de salvar seu estado e enviar ao(s) backup(s), o que consome processamento e largura de banda
- Transferência de estado pode ser incremental
- Obs: Quanto maior o intervalo entre *checkpoints*, menor a sobrecarga no primário, e maior o tempo de recuperação de falhas

# Replicação

## Replicação Ativa

Um grupo de réplicas de um componente recebe uma requisição de um cliente

Todas as réplicas processam a requisição concorrentemente e enviam as suas respostas ao cliente

Não é preciso sincronizar o estado das réplicas, pois todas executam os mesmos procedimentos

# Replicação

## Replicação Ativa

# Replicação

## Replicação Ativa

- O cliente precisa de apenas uma resposta
- A resposta válida para o cliente pode ser:
  - A primeira recebida
  - A mais freqüente (votação)
    etc.
- Com isso, a replicação ativa pode tolerar faltas de valor por meio de votação
  - Em 2N+1 réplicas podemos ter N respostas erradas sem ocasionar falha do sistema

# Replicação

Replicação Ativa

Considerações:

Alto custo para execução das réplicas ativas
As requisições devem ser entregues na mesma ordem para todas as réplicas à usar protocolo de difusão atômica

Ordenação de mensagens tem custo alto

Recuperação é mais rápida que na replicação passiva, pois caso uma réplica falhe, as demais continuam funcionando normalmente

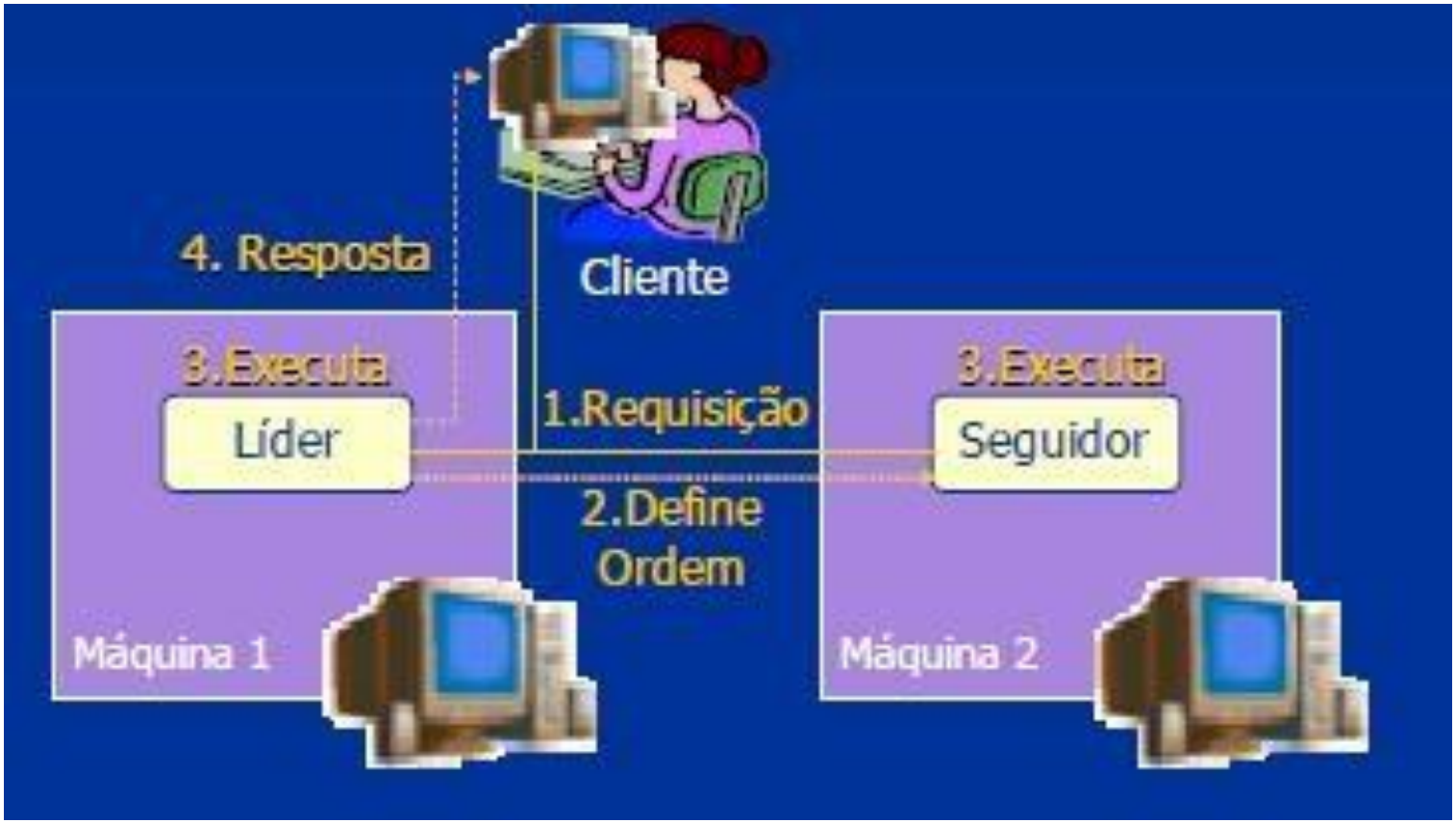
# Replicação

## Replicação Semi-Ativa (Líder-Seguidores)

- Um componente (líder) possui uma ou mais réplicas (seguidores)
- Cada requisição é enviada a todos, que as executam na ordem definida pelo líder
- Apenas o líder responde ao cliente que efetuou a Requisição
- Não é preciso sincronizar o estado das réplicas, pois todas executam os mesmos procedimentos

# Replicação

## Replicação Semi-Ativa (Líder-Seguidores)



# Replicação

## Replicação Semi-Ativa (Líder-Seguidores)

### Considerações:

Mesmo que as requisições dos clientes sejam entregues fora de ordem, todas as réplicas chegarão ao mesmo estado, já que a ordem de execução é arbitrada pelo líder

O tempo gasto para processamento nas réplicas seguidoras é grande, já que elas também têm que processar a chamada

Caso o líder falhe, um seguidor é escolhido para assumir o seu lugar

# Replicação

## Replicação Preguiçosa (Lazy)

As operações são recebidas por qualquer uma das réplicas que devem:

Atender a requisição
Enviar a resposta ao cliente
Difundir em background a requisição para as outras réplicas

# Replicação

## Replicação Preguiçosa (Lazy)

### Operação com ordem causal

# Replicação

## Replicação Preguiçosa (Lazy)

### Considerações:

Reduz a sobrecarga na execução das operações, já que não é necessário usar um protocolo de difusão atômica

Exige que a semântica das operações seja condizente com uma ordenação causal

Ex: Session Stickness para aumentar a localidade de referência espacial

# Detecção de Falhas

A falha de um componente de um sistema pode levar todo o sistema a falhar

Mesmo que o sistema consiga tolerar a falha do componente, este deve ser recuperado para restaurar a capacidade do sistema de tolerar faltas

> Ex.: na replicação passiva, se o único backup existente assume o lugar do primário, é preciso criar um novo backup

É preciso detectar as faltas sofridas pelos componentes para poder recuperá-los

# Detecção de Falhas

## Detecção Local de Falhas

Podem ser usados diversos métodos:

- Rotinas de auto-verificação (*self-check*)
- Guardiões: verificam constantemente as saídas geradas por um componente
- *Watchdogs* : componente deve constantemente reiniciar um temporizador antes que ele se esgote, indicando uma falha

Problema: mesmo para um observador local, processos lentos podem parecer falhos

# Detecção de Falhas

## Detecção Distribuída de Falhas

Um componente do sistema envia mensagens periodicamente aos seus pares e avisa que está vivo (I am alive) ou pergunta se eles estão vivos (Are you alive?)
Se um componente não se manifestar por um determinado tempo, ele é suspeito de falha
Suspeitas infundadas podem ser causadas por atraso, particionamento ou falha da rede

# Detecção de Falhas

## Diagnóstico do Sistema

Componentes faltosos podem reportar erroneamente o estado dos seus pares

Em um sistema com *f* componentes faltosos, cada componente deve ser testado por pelo menos *f* outros, e precisamos de $n \geq 2f + 1$ elementos para detectar corretamente a falta

Para diagnosticar falhas de componentes do sistema, um elemento deve coletar e analisar os dados obtidos dos demais componentes

# Detecção de Falhas

## Detector de Falhas

- Serviço ou módulo que verifica a ocorrência de falhas em componentes do sistema
- Implantado junto ao componente
- Executa um algoritmo de detecção de falhas
- Interage com detectores de outros componentes do sistema
- O componente pode requisitar ao seu detector informações sobre o estado de outros componentes do sistema

# Detecção de Falhas

## Tipos de Detectores de Falhas

### Perfeitos

- Determinam precisamente se um componente do sistema falhou ou não
- Todos os componentes têm a mesma visão

### Imperfeitos

- Detectores determinam se um processo é suspeito de falha ou não
- Diferentes componentes podem ter visões distintas de um mesmo componente

# Detecção de Falhas

## Tipos de Detectores de Falhas

- Detectores perfeitos são difíceis de obter, principalmente em sistemas distribuídos
- Detectores quase-perfeitos podem ser obtidos usando *crash* controlado
  - Se um componente é suspeito de falha, ele é removido do sistema
  - O componente passa a ser ignorado por todos os demais componentes
  - Pode levar a descartar componentes que estão funcionando corretamente

# Recuperação de Falhas

## Recuperação de Erros

Ao perceber um erro, o componente pode tentar recuperar-se automaticamente

Recuperação de erro por retrocesso (*backward error recovery*): componente volta a um estado anterior ao erro e continua ativo

Exemplos: reinicia a execução de um método, retransmite pacotes perdidos, etc.

Operações posteriores ao instante de retrocesso são perdidas, mas seu efeito pode ainda ser sentido no sistema, levando possivelmente a inconsistências

# Recuperação de Falhas

## Recuperação de Erros (cont.)

- Recuperação por avanço (*forward error recovery*): componente toma medidas que anulem ou aliviem o efeito do erro e continua a operar normalmente
  - Exemplo: descartar pacotes, substituir um valor inválido pelo valor válido anterior, etc.
    Usada quando não há tempo para voltar para estado anterior e retomar execução, ou quando ações não podem ser desfeitas

# Recuperação de Falhas

## Recuperação de Falhas

- Se ocorrer a falha de um componente, um sistema tolerante a faltas deve mascará-la usando as réplicas disponíveis
- Na replicação passiva, substituir o primário por um backup e criar um novo backup
- Na replicação semi-ativa, substituir o líder por um de seus seguidores e criar um seguidor
- Nas replicações ativa e lazy, criar uma nova réplica para manter a capacidade do sistema de tolerar faltas (ou falhas de componentes)