Sumário

- Gestão de dados replicados.
- Controlo de cópias.
- Controlo da concorrência.
- Algoritmos estáticos e dinâmicos.

Motivação

- Melhorar o desempenho.

  Permitindo que os clientes acedam a dispositivos mais "próximos" (em tempo de acesso) ou menos sujeitos a congestão.

- Tolerância a faltas.

  Permite aumentar a disponibilidade e fiabilidade dos dados na presença de faltas temporárias ou permanentes nos dispositivos.

Modelos de coerência

- Uma vez que os dados vão estar replicados.

  É necessário definir qual o comportamento correcto do sistema.

  Este aspecto já foi abordado quando se falou de memória partilhada.

- Vamos dar ênfase a algoritmos que (em conjungação com suporte transacional) permitem garantir a serializabilidade.

Gestão da replicação

- Controlo de cópias.

  Que permite reconhecer qual a cópia mais actual.

- Controlo da concorrência.

  Ordenação dos vários acessos aos dados.

Controlo de cópias

- Nem todas as cópias necessitam de manter os valores actualizados desde que os utilizadores obtenham os valores desejados.
- Geralmente, o controlo de cópias é feito numerando versões.

  Sempre que os dados são actualizados incrementa-se a versão.

  Quando se acede a várias cópias o número de versão permite reconhecer qual o valor mais actual.

Controlo da concorrência

- Na presença de replicação, o acesso aos dados pode ter de ser executado em mais que uma réplica.
- Dois acessos interseptam-se quando são feitos a dois conjuntos de réplicas cuja intersepção é não nula.
- Usualmente, a intersepção ocorre em pelo menos uma cópia.

  Compatibilidade verificada através de um mecanismo de controlo da concorrência local (ex: trincos).
- Quando se sabe de antemão que duas operações são compatíveis, este requisito pode ser enfraquecido.

  Por exemplo, não é necessário que duas operações de leitura se intersep-tem.

Duas classes de algoritmos

- Algoritmos não-adaptativos (ou estáticos).

  Os algoritmos estáticos são configurados durante a iniciação do sistema.

  Não alteram o seu funcionamento no caso de ocorrência de faltas.
- Algoritmos adaptativos (ou dinâmicos).

  Os algoritmos dinâmicos são capazes de se adaptar a mudanças no ambiente, tais como falhas ou partições.

  Mais eficientes.

  Exigindo um suporte sofisticado (detecção de faltas).

Algoritmos estáticos

- O algoritmo é sempre o mesmo.
- Deve funcionar para qualquer cenário.

  Falha de réplica, partição na rede, etc.
- É configurado na iniciação do sistema.

Algoritmo simples

- Escreve em todas as réplicas, lê apenas de uma.

  As operações interseptam-se sempre.

  Escrita dispendiosa (e só funciona quando todas as réplicas acessíveis).

- Qualquer réplica contém sempre o estado actualizado.

  Leituras muito eficientes.

Votação ponderada

- A cada réplica é atribuido um voto.
- A cada operação é associado um quorum.

  A soma do quorum de duas operações incompatíveis, deve exceder o número de votos no sistema.

- Definem-se o votos em função das propriedades das réplicas e o quorums em função dos padrões de utilização.

Coutadas ("coteries")

- Os quorums são definidos indicando explicitamente (vários) conjuntos de réplicas que devem ser acedidas.

  A operação só tem sucesso caso se consiga aceder a todas as réplicas em um dos conjuntos associados à operação.

- Os conjuntos associados a operações incompatíveis devem sempre interseptar-se.
- Existem coutadas que não se conseguem exprimir através de votação.

Coutadas: desvantagens

- Mais complexo do que a votação:

  É necessário manter e consultar a lista de conjuntos de quorum associados à operação.

- Difícil de seleccionar um conjunto de coutadas adequado.

Representações estruturais

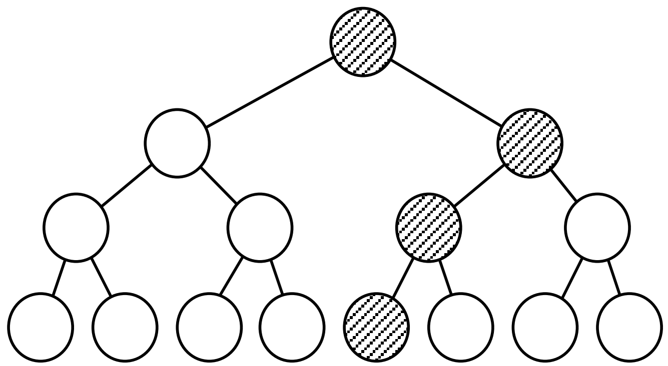
- Utilizar estruturas lógicas, tais como árvores, grelhas (*grids*) definir coutadas.
- Mais simples de "visualizar".
- Mais fácil de definir algoritmos de verificação.

Algoritmo em árvore

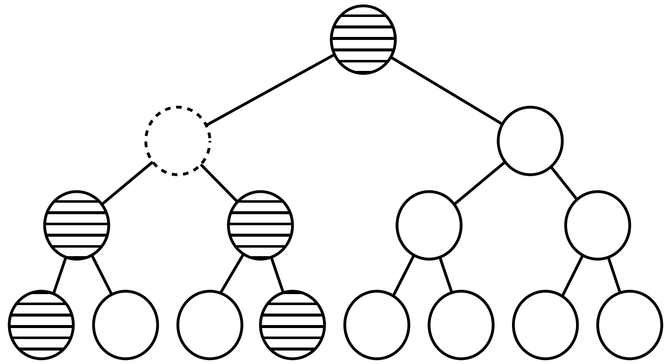
- Organiza logicamente as cópias numa árvore binária.
- O algoritmo tenta obter um *quorum* seleccionando um caminho a partir da raiz até uma das folhas.

  A raiz entra em todos os quorums e fica sobrecarregada.
- Se nenhum destes caminhos for encontrado devido à inacessibilidade de uma dada cópia c, essa cópia é substituida no *quorum* por dois caminhos, cada qual começando num nó filho do nó correspondente à cópia inacessível (o algoritmo é recursivo).

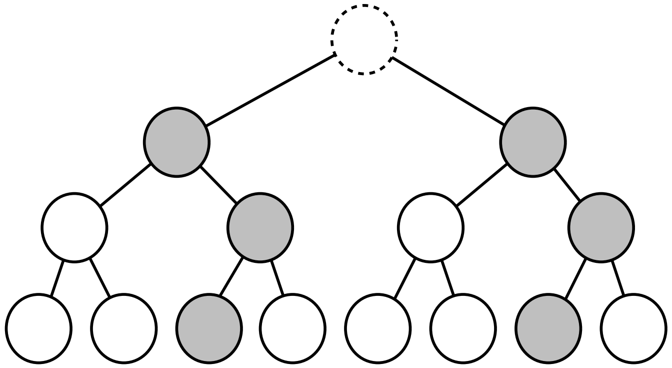
Algoritmo em árvore: exemplo 1



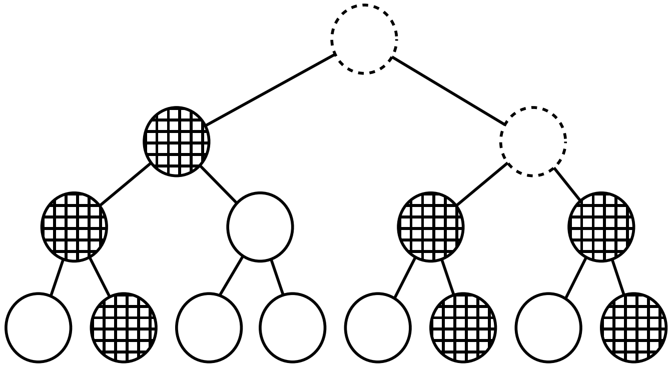
Algoritmo em árvore: exemplo 2



Algoritmo em árvore: exemplo 3



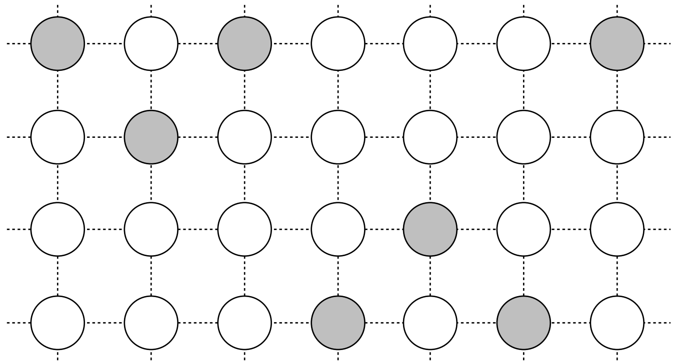
Algoritmo em árvore: exemplo 4



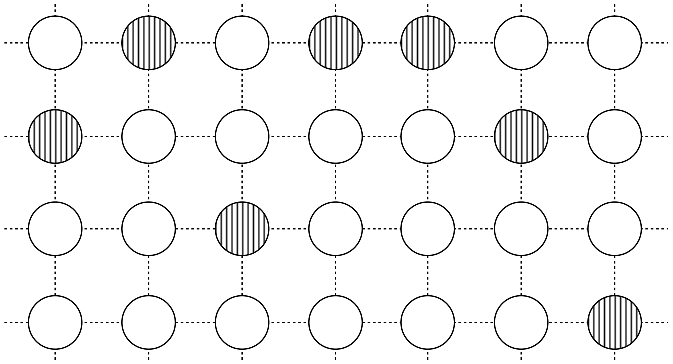
Algoritmo em grelha

- Organiza logicamente as cópias numa grelha rectangular.
- Um *quorum* para leitura deve conter uma cópia de cada coluna.
- Um *quorum* para escrita deve conter, para além de um *quorum* de leitura, todos os elementos de uma coluna da grelha.

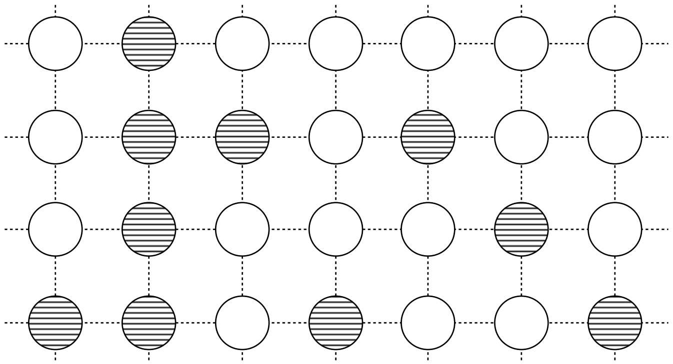
Algoritmo em grelha: leitura 1



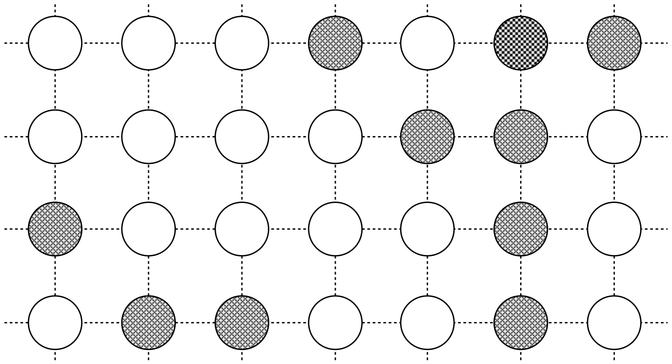
Algoritmo em grelha: leitura 2



Algoritmo em grelha: escrita 1



Algoritmo em grelha: escrita 2



Algoritmos dinâmicos pessimistas

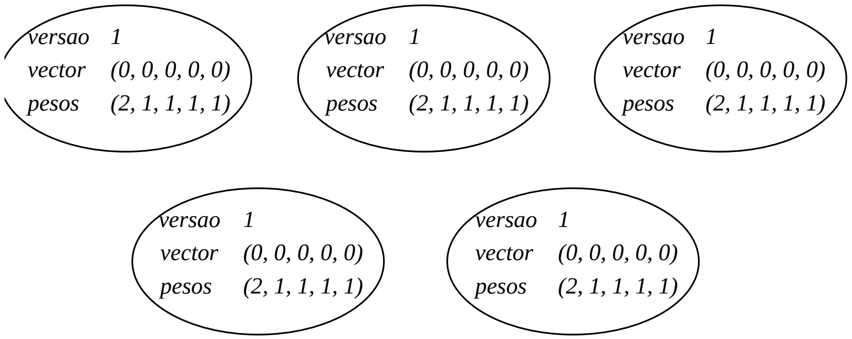
- Um algoritmo dinâmico tenta garantir o progresso recolhendo informação acerca do estado do sistema e reconfigurando-se em tempo de execução.
- Tipicamente, os sistemas dinâmicos tentam trabalhar com pequenos *quorums* para leitura, uma vez que na maioria dos sistemas esta é a operação dominante.
- Dinamicamente, o sistema muda os conjuntos de *quorum*, escolhendo *quorums* que favoreçam o progresso do sistema.

Votação ponderada dinâmica

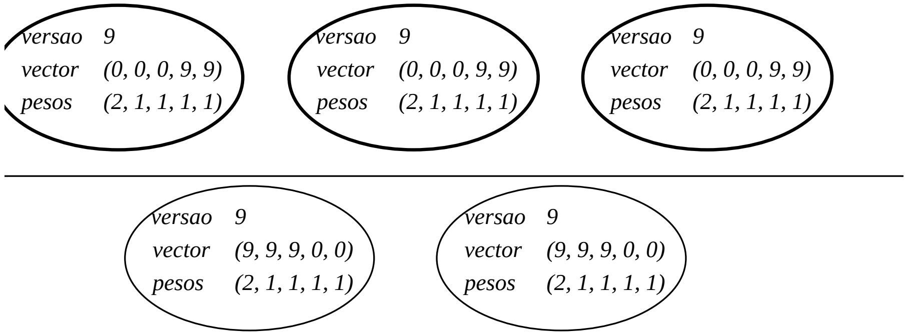
- Altera o critério de maioria quando suspeita da existência de uma partição.
- Sempre que este critério é alterado, memoriza-se a versão que cada uma das cópias possuia nesse momento.

  Esta informação designa-se vector de partição.
- Só se permite que o critério seja alterado quando se consegue aceder a uma maioria das cópias actualizadas.

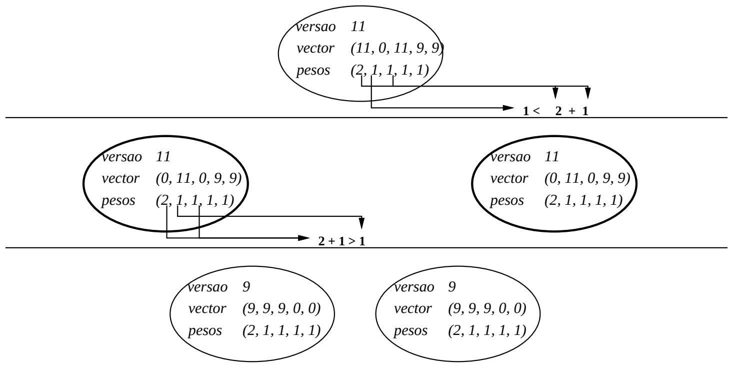
Exemplo: início



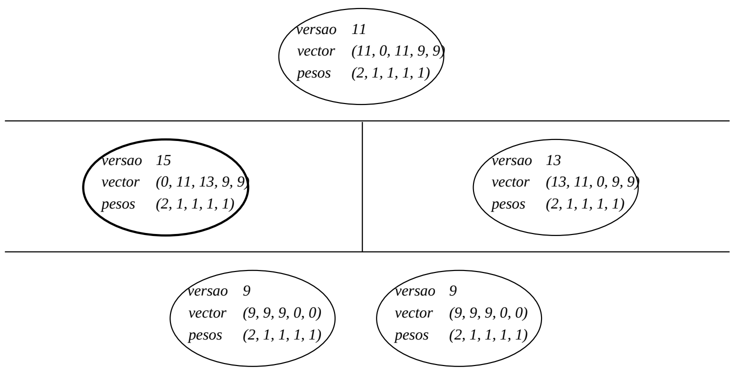
Exemplo: partição



Exemplo: nova partição



Exemplo: (aínda) mais outra partição

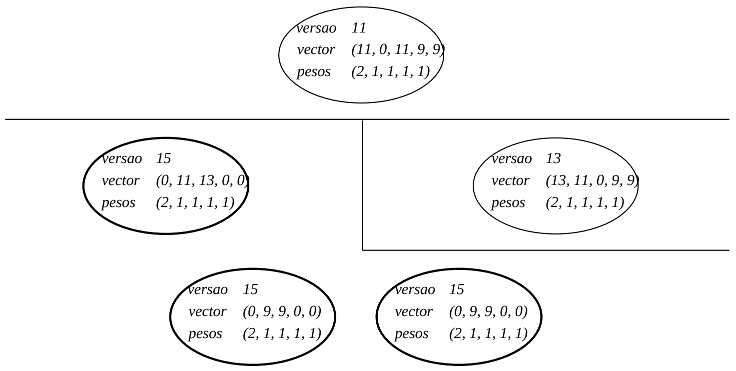


Reunião

- Quando a conexidade é retomada, é necessário actualizar as réplicas.
- Isto é feito através de um procedimento de união de partições.

  Durante a execução deste procedimento, o acesso aos dados é temporariamente inibido.

Exemplo: reunião



Primário-secundário

- Todas as operações incompatíveis são realizadas numa única cópia (designada por primário).
- O primário propaga as informações para as restantes cópias (secundários).
- O primário pode mudar em tempo de execução.

  Passagem de testemunho.

- Em caso de falha do primário, elege-se um dos secundários para novo primário.

  Garantir que apenas um primário está activo.

Cópias disponíveis

- Pressupõe-se que uma cópia inacessível falhou.
- Continua-se sempre com as cópias disponíveis.

  “e possível operar com uma única cópia.
- As cópias ao recuperarem, necessitam de contactar a cópia activa.
- Se falharem todas as réplicas, é necessário esperar pela recuperação da última cópia a falhar.

Utilizando grupos

- Utilização de comunicação em grupo para aceder às replicas.
- Bastante fácil concretizar um algoritmo dinâmico.

  Escreve em todos os elementos do grupo, lê de uma réplica qualquer.

  O protocolo de filiação mantém informação acerca de quais as réplicas acessíveis.
- Partição primária ou cópias disponíveis consoante o serviço de filiação tolera (ou não) partições.

Resumo

- Numeração de versões.
- Votação ponderada.
- Coutadas.
- Vectores de partição.
- Cópias disponíveis.
- Usando grupos.