2007 - 11 junho

Parte A

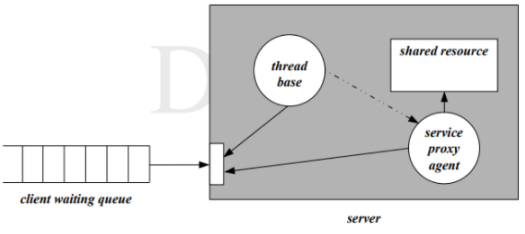
1- Explique o que entende por sistema distribuído. Dê dois exemplos da sua aplicação sobre plataformas hardware com características distintas. Qual a motivação da sua construção em cada um dos casos ?

A definição do sistema distribuído afirma que se está a lidar com um sistema cujo os componentes, localizados em diferentes nós de processamento de um sistema de computador paralelo, comunicam e coordenam as suas ações por passar mensagens. Duas formas da sua aplicação é:

- Sistemas de Clusters - que consiste em uma coleção de sistemas de computador interconectado por uma rede de alta velocidade, tendo como objetivo é a paralização de aplicativos. Este funciona de forma a que os sistemas de computador sejam idênticos e apenas há um único nó mestre que lida com a alocação de nós de processamento para um programa específico e então mantém uma fila de trabalhos enviados e interfaces com usuários do sistema.
- Sistemas de grade - o motivo principal é para reunir os recursos de diferentes organizações para que uma comunidade de pessoas possam cooperar, é então formada uma organização virtual , onde apenas os membros têm direitos de acesso aos recursos fornecidos

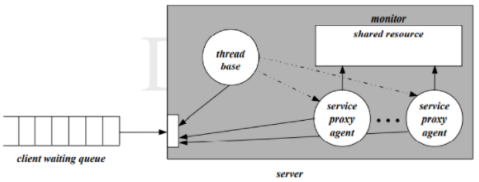
2 - Descreve esquematicamente, introduzindo comentários apropriados sobre a sua operacionalidade, vantagens e inconvenientes, as três variantes do modelo cliente-servidor que foram estudadas.

Variante do tipo 1(solicitação de serialização) - apenas um processo do cliente é atendido de cada vez, o que significa que o thread base, após receber uma solicitação de conexão instância um agente proxy de serviço e espera que acabe(feche) antes de começar a ouvir novamente. Dado que apenas atente um processo de cada vez e que apenas um agente proxy de serviço está ativo, nenhuma segurança é necessária para garantir exclusão mútua na sessão de acesso. Mas dado isto, é uma variante bastante ineficiente pois :



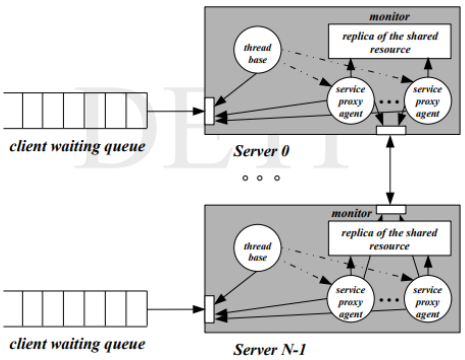
- o tempo de atendimento não é minimizado, dado que não aproveita os tempos mortos de interação , devido a falta de competição
- dá origem a uma espera ocupada na tentativa de sincronizar vários clientes que processam no mesmo recurso partilhado

Variante do tipo 2(Replicação do servidor) - os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo. Esta é a forma tradicional de como os servidores são configurados, de forma a que se aproveite ao máximo os recursos do sistema do computador onde o servidor está localizado ao:



- o tempo de atendimento é minimizado, pois se aproveita a interação de tempos mortos através da simultaneidade
- permite a sincronização de diferentes processos do cliente no mesmo recurso compartilhado.

Variante tipo 3(Replicação de recursos) - o serviço é disponibilizado simultaneamente em múltiplos sistemas, cada um executando a variante do tipo 2(na qual os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo.), então o recurso partilhado é replicado em cada servidor o que resulta em múltiplas cópias. Este é um modelo sofisticado que tem como objetivo maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga(potencializando assim a escalabilidade)



- o serviço é mantido operacional contra a falha de servidores específicos
- as solicitações do cliente são distribuídas entre os servidores disponíveis usando uma política, pré-fornecido pelo serviço DNS, de

associação geográfica para solicitações globais e de associação rotativa para solicitações locais
- quando há uma alteração de dados locais em uma das réplicas do recurso compartilhado, a necessidade de manter as diferentes réplicas consistentes surge e deve ser tratado.

*3 - No contexto do RMI em Java, explique o que entende por objectos [remotos] estáticos e dinâmicos. Descreva funcionalmente como é que eles podem ser implementados e dê um exemplo de cada tipo da sua aplicaçã*o.

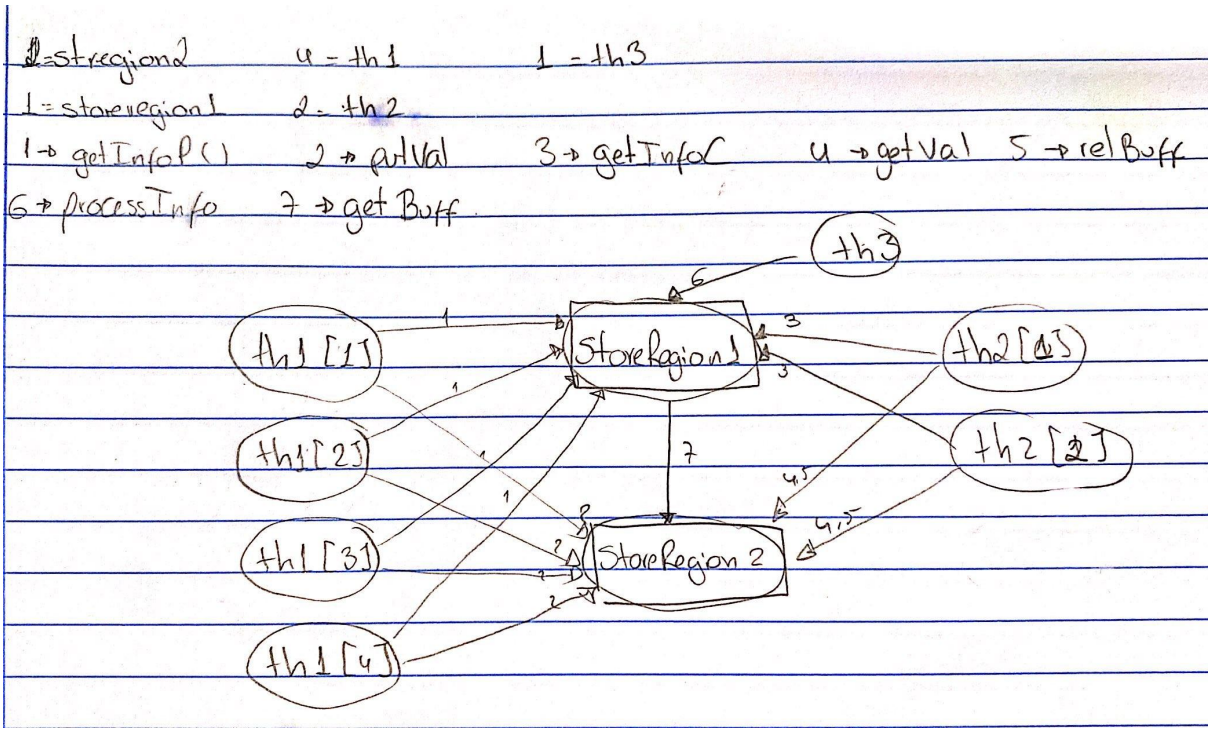
Um objeto estático (cave) tem um tempo de vida igual ao da thread que o instância, implementada na região de dados partilhada.
Enquanto o objeto dinâmico (party) tem o tempo de vida inferior ao thread, o que lhe permite instanciar múltiplos objetos, implementa um canal de comunicação para permitir a comunicação entre entidades existentes.

4 - No âmbito da comunicação entre pares, descreva detalhadamente um algoritmo que permite de uma forma distribuída e dinâmica a um dado processo assumir-se como líder. Indique claramente quais são os pressupostos que estão subentendidos.

Assumindo-se que não existem falhas na rede, cada um dos nós no início gera um número de identificação para si mesmo, e cada nó assume-se a si mesmo como líder, marca a mensagem como sua e passa-a ao nó seguinte, em sentido do relógio.
Após a receção da mensagem o nó seguinte, verifica o número de id. do líder e verifica se o seu número de identificação é maior, se for maior, substitui o número de identificação do líder na mensagem pelo seu, se não for maior não troca e reenvia a mensagem para o próximo nó.
As mensagens passam por todos os nós, até que as mensagens cheguem aos seus nós iniciais. Aqui cada um dos nós verifica a sua mensagem e vê se o número de identificação do líder da sua mensagem , caso seja igual, esse nó identifica-se como líder, se não for igual, identifica-se como sendo um coordenador.

Parte B

1 - Represente as entidades activas por círculos e as entidades passivas por retângulos, faça um diagrama ilustrativo em presença e indique por palavras simples qual é o papel desempenhado pelas threads de cada tipo (não mais que uma frase)



A thread1 vai buscar o valor guardado no storeRegion1 e põe o valor na storeRegion2 com o id da thread1 em questão. A thread2 vai buscar o valor que está guardado na storeRegion1 e o valor que está guardado na storeRegion2 e faz print destes valores, depois liberta o buffer na storeRegion2. A thread3 processa a informação que está na storeRegion1.

2008 - 20 Junho

Parte A

1 - Explique qual é a importância de se garantir a tolerância a falhas no âmbito dos sistemas distribuídos e indique quais são as estratégias mais comuns usadas no tratamento das falhas. Dê um exemplo de cada uma delas.

Pag 157/158 ? Failure Handling ? variantes?

- mascarando as falhas - algumas das falhas detectadas podem ser ocultadas ou, pelo menos, seu efeito seja tornado menos severo

    - mensagens perdidas, se detectadas, podem ser transmitidas novamente em muitos casos
    - a replicação de recursos pode permitir a proteção das informações armazenadas quando há corrupção de dados em algumas das cópias
- recuperação de falhas - para que seja possível, é necessário projetar os componentes de software de forma que os estados internos dos processos envolvidos são armazenados periodicamente; quando ocorre uma falha, o processo é reiniciado a partir do último estado salvo na prática, isso também pode exigir a migração de código para outros recursos de hardware e o uso de cópias atualizadas dos dados
- tolerar falhas - algumas falhas devem ser toleradas; qualquer tentativa de resolver são inúteis ou impraticáveis; nesses casos, em vez de deixar o usuário encalhado, enquanto sucessivas tentativas de acesso são realizadas, é melhor deixar ele / ela saber do fato.
-

2 - Descreva as operações principais que têm que ser efetuadas para comunicação, na perspectiva do programador, na implementação de um modelo cliente-servidor, com replicação do servidor, usando o protocolo TCP, quer do lado do servidor, quer do lado do cliente.

3 - Indique quais são as diferenças fundamentais que existem em Java, em termos de transparência, no acesso a objetos locais e a objetos remotos.

O método de invocação remota(RMI) permite que haja interação de programas que estão a ser executados em diferentes nós de um sistema distribuído.(pelas RPC - Remote Producer Calls)
Com o RMI a transparência de objetos é conseguida uma vez que o nível de abstração permite invocar objetos remotos da mesma maneira que se invocam objetos locais.
existem 3 características a ter em atenção , pois uma RMI não funciona exatamente da mesma forma que a chamada de procedimento local:

- A chamada pode falhar, pois pode ainda não estar instalada, ou a infraestrutura de comunicação não estar a funcionar corretamente.

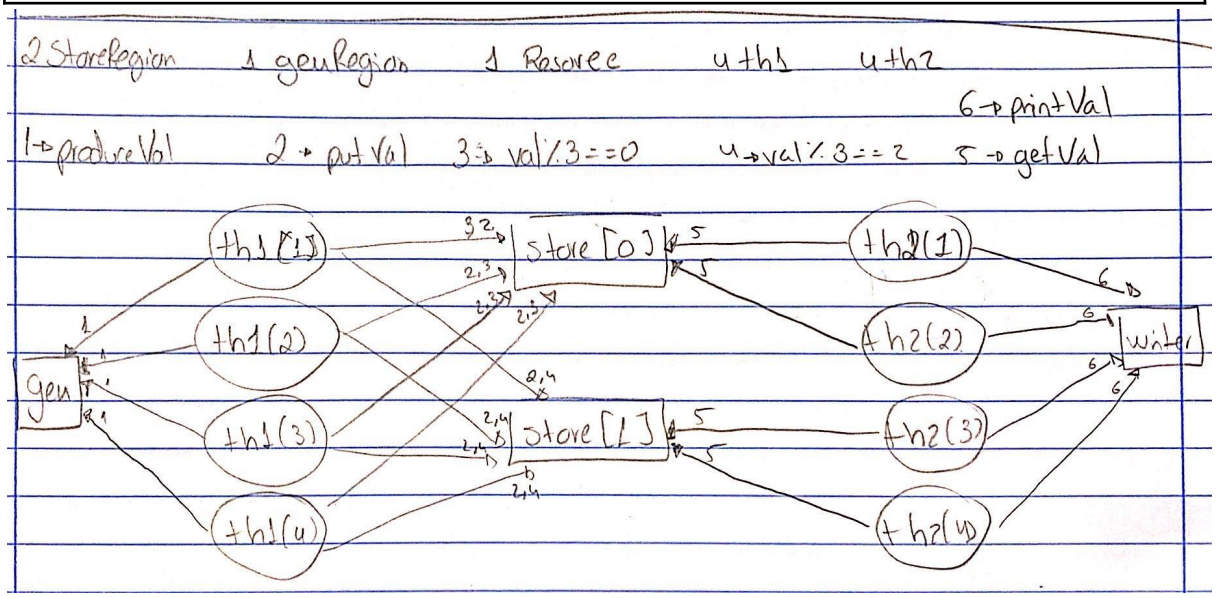
- Os parâmetros do procedimento e de valores de return devem ser passado por valor , pois como estão em diferentes endereços, o meio de comunicação entre eles é por passar dados relevantes juntamente com o seu formato e estrutura.
- e o procedimento de execução remota demora mais tempo que a execução de procedimentos local.

4 - No âmbito da comunicação entre pares, descreva detalhadamente um algoritmo que permita de uma forma distribuída e dinâmica fazer-se a sincronização dos relógios locais. Que tipo de ordenação de acontecimentos é que resulta da sua aplicação. Mostre como é possível garantir a consistência de dados entre diferentes cópias localizadas em locais geograficamente separados quando ele é usado.

- Lamport propôs o seguinte algoritmo para resolver o problema
    - cada processo p i , ao afirmar que a próxima operação irá modificar o valor de um registro de sua cópia local, constrói uma mensagem com todos os dados relativos à operação e anexa um carimbo de hora com o valor de sua marcação de relógio local do evento
    - a mensagem é enviada a todos os membros do grupo, ela própria incluída
    - ao receber a mensagem, cada processo p i ajusta seu relógio local de acordo às regras prescritas por Lamport e insere a mensagem em uma fila local em ordem crescente de seu carimbo de tempo estendido
    - uma mensagem de reconhecimento é enviada a todos os membros do grupo, incluindo ela própria
    - as operações descritas nas mensagens armazenadas em cada fila local são executadas por cada processo p i na ordem pré-estabelecida quando todos os membros do grupo reconheceram a operação.

Parte B

1 - Represente as entidades activas por círculos e as entidades passivas por retângulos, faça um diagrama ilustrativo em presença e indique por palavras simples qual é o papel desempenhado pelas threads de cada tipo (não mais que uma frase)



As threads do tipo 1 vão produzir um valor que vão buscar a GenRegion e vão guardar esse valor no StoreRegion, o valor é guardado dependendo desse mesmo valor, se o valor ao dividir por 3 tiver resto de zero é guardado no store[0] se tiver resto de 2 fica guardado no store[1]. Depois então as threads do tipo 2 vão buscar os valores a StoreRegion dado que as threads com ids 1 e 2 vão buscar a store[0] e as 3 e 4 a store[1], depois de ter o valor vão imprimir os resultados no Resource (writer).

2 - Assuma que, quando o programa é executado, se obtém o resultado seguinte:

O valor processado por 1 e por 2 foi 24.
O valor processado por 3 e por 1 foi 3.
O valor processado por 1 e por 2 foi 12.
O valor processado por 2 e por 4 foi 5.
O valor processado por 4 e por 3 foi 4.
O valor processado por 4 e por 4 foi 11.
O valor processado por 3 e por 2 foi 36.
O valor processado por 1 e por 3 foi 2.
O valor processado por 4 e por 3 foi 7.
O valor processado por 2 e por 1 foi 30.

O valor processado por 4 e por 4 foi 16.
O valor processado por 1 e por 1 foi 18.

Tenha em atenção que, face à aleatoriedade introduzida, este não é o único resultado possível. De facto, nem sequer está correto. Existem três erros, respectivamente, nas linhas 2, 6 e 9. Identifique-os. Justifique cuidadosamente a sua resposta.

O primeiro erro é na 2ª linha : *O valor processado por 3 e por 1 foi 3.* pois dado que 3%3 == 0 o resultado do valor processado deveria de ser 6. E não existe nenhum valor que *2 de 3 por isso está incorreta esta linha.
O segundo erro é na 5ª linha: *O valor processado por 4 e por 3 foi 4.* Dado que o valor 4 ao dividir por 3 dá resto de 1 logo não é guardado na store.
O terceiro erro é na 9ª linha: *O valor processado por 4 e por 3 foi 7.* Dado que o valor 7 ao dividir por 3 dá resto de 1 logo não é guardado na store.

3 - Explique como é sempre garantida a terminação do programa.

Dado que ao criar o objeto writer do Resource é definido um valor máximo = 12 que é o número de números disponíveis na GenRegion, e cada vez que é imprimido um número é retirado a esse valor máximo 1 unidade. Ao chegar a 0 esse valor, faz com que acordem a storeRegion e ponha o stat=2 que faz com que que acabe com o while do getVal e então é retornado um true ao writer para que saiba que acabou o processo.

2012 - 21 Junho

Parte A

1 - Foi referido que a motivação principal para a construção e utilização de sistemas distribuídos é a partilha de recursos. Esta partilha manifestando-se principalmente na paralelização de operações e na disponibilidade de serviços. Apresente exemplos, um de cada tipo, que justifiquem a afirmação anterior.

Como foi dito anteriormente, a partilha de recursos foi o motivo a que levaram com que se começa-se a utilizar os sistemas distribuídos, esta partilha de recursos é preferível devido a 2 acontecimentos, um deles sendo a

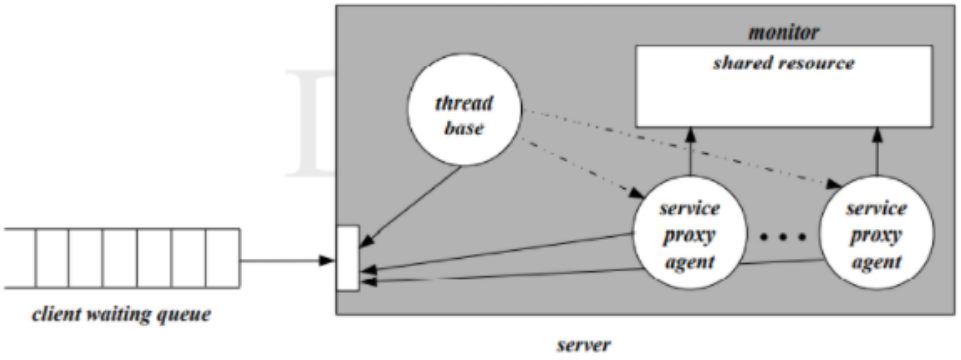
paralelização pois tira vantagem de vários processadores e outros componentes hardware de um sistema de computador paralelo para obter uma execução mais rápida e eficiente de uma aplicação, sendo o outro acontecimentos a disponibilidade de serviços que fornece uma funcionalidade bem definida a um grupo ampliado de aplicações de uma forma consistente, confiável e segura.

2 - Considere a troca de mensagens como o paradigma de comunicação que foi escolhido para se estabelecer um modelo cliente-servidor entre processos residentes em diferentes nós de processamento de um sistema distribuído e admita a replicação do servidor (serviço em concorrência de clientes distintos) e que foram usados na sua implementação os Java sockets.
Face a este enquadramento, responda justificadamente às questões seguintes:

i. *Apresente um diagrama esquemático que descreva a interação funcional entre os diferentes componentes de um tal sistema no lado do servidor, identificando cada um dos componentes introduzidos.*

ii. *O mecanismo de replicação do servidor não pode ser instanciado indefinidamente. Fatores limitativos determinantes são os recursos do sistema computacional onde o servidor está instalado. Identifique dois deles e explique qual é o seu impacto.*

i.



ii. O mecanismo de replicação do servidor não pode ser instanciado indefinidamente devido a fatores limitativos que são a exclusão mútua no acesso a região partilhada que faz com que o recurso partilhado se torne um monitor para que seja assegurada a exclusão mútua e a sincronização pois os recursos precisam de se comunicar entre si para sincronizar os dados. (not sure, inventei um pouco)

3 - Explique como se efetua o acerto dos relógios locais entre os diferentes nós de processamento de um sistema distribuído usando relógios de tipo escalar (Lamport) .

Lamport mostrou que os relógios locais não precisam de ser estritamente iguais no seu ritmo, mas que o que realmente interessava era que mantivessem a ordem a que aconteciam da mesma forma nos sistemas distribuídos que estivessem envolvidos de alguma forma. Então ele definiu um evento a qualquer atividade relevante que ocorre durante a execução de um processo. Dado isto a ordem dos eventos é mantida segundo :

- quando 2 processos ocorrem no mesmo processo, eles são feitos pela ordem percebida pelo processo
- e quando uma mensagem é trocada entre processos, o evento do envio da mensagem tem de ficar necessariamente posta **antes** do evento de receber essa mesma mensagem.

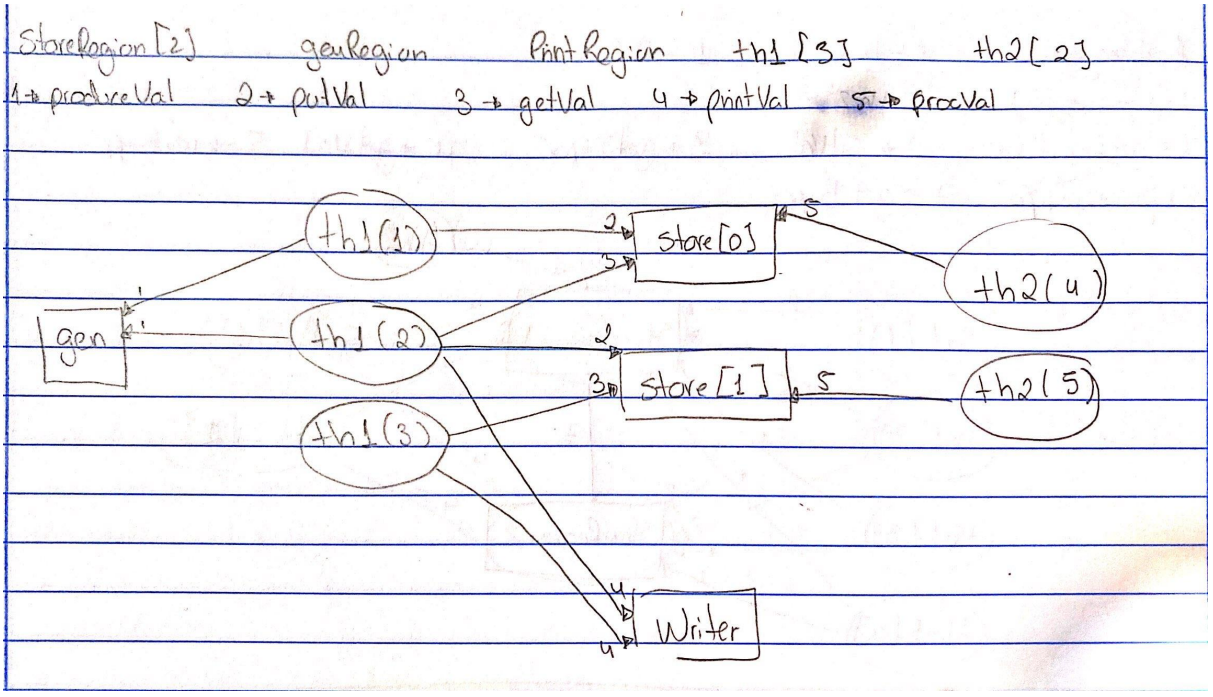
4 - Considere uma situação no contexto da comunicação entre pares em que coexistem 3 processos P1, P2 e P3, que pretendem eleger um deles como chefe. Descreva os passos principais de um algoritmo que conduz a essa eleição. Admita que não há perda de mensagens.

Assumindo-se que não existem perdas de mensagens, cada um dos nós P1,P2 e P3 no início geram um número de identificação para si mesmo, e cada nó assume-se a si mesmo como líder, marca a mensagem como sua e passa-a ao nó seguinte, em sentido do relógio.
Após a receção da mensagem o nó seguinte, verifica o número de id. do líder e verifica se o seu número de identificação é maior, se for maior, substitui o número de identificação do líder na mensagem pelo seu, se não for maior não troca e reenvia a mensagem para o próximo nó.

As mensagens passam por todos os nós, até que as mensagens cheguem aos seus nós iniciais. Aqui cada um dos nós verifica a sua mensagem e vê se o número de identificação do líder da sua mensagem , caso seja igual, esse nó identifica-se como líder, se não for igual, identifica-se como sendo um coordenador.

Parte B

1 - Represente as entidades activas por círculos e as entidades passivas por retângulos, faça um diagrama ilustrativo em presença e indique por palavras simples qual é o papel desempenhado pelas threads de cada tipo (não mais que uma frase)



As threads de tipo 1 tem diferentes objetivos conforme os seus ids, a thread1(1) vai ao genRegion buscar um valor (produzir um valor) e vai guardar esse valor na store[0], a thread1(2) vai também produzir um valor na genRegion e vai guardar esse valor na store[1], e depois vai então buscar o valor ao store[0] (inserido pela thread1(1)) e vai po-lo também na store[1]. a thread1(3) vai buscar o valor ao store [1] (introduzido pela thread1(2)) e vai imprimi-lo no writer. As threads do tipo 2 servem para garantir que tudo está impresso e que já pode fechar o processo , a thread2(4) verifica a store[0] e a thread2(5) verifica a store[1].

3 - Quais são os números mínimo e máximo de linhas de texto imprimidas por uma execução do programa na versão apresentada ? Justifique devidamente a sua resposta.

O número máximo de vezes e mínimo é de 6 linhas, pois apenas quando o genRegion manda o val = 0 é que acaba o programa e ele apenas envia o 0 quando foram enviados tantos números como os que estão no valSet, dado que são 6 valores.

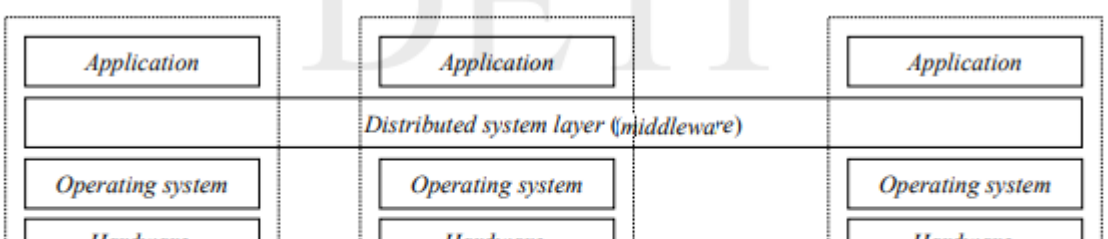
2016 - 21 Junho

Parte A

1 - *O que se entende por camada de middleware? Faça um diagrama ilustrativo que sinalize a presença da camada para o caso de uma máquina paralela de acoplamento solto, formada por três nós de processamento, que executa uma aplicação distribuída. Explique porque é que a máquina virtual de Java (JVM) pode ser considerada um componente do middleware.*

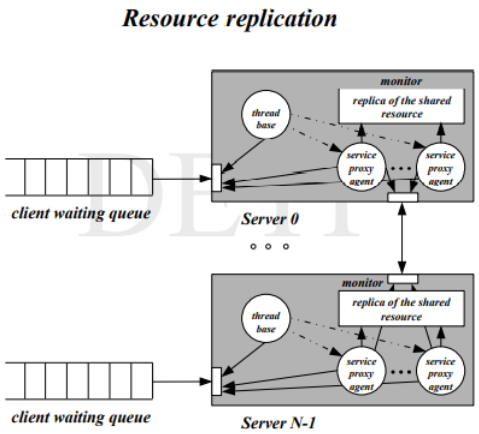
Middleware pode ser descrito como um conjunto de processos que se comunicam entre si de forma a mascarar as heterogeneidades dos sistemas operativos de tal modo a que a aplicação possa ser executada em todos eles.>
A máquina virtual de Java (JVM) é constituída por uma camada chamada middleware que faz com que as aplicações sejam totalmente independentes da plataforma hardware e do Sistema operativo, implementado uma tripla camada de segurança que protege o sistema contra código não-confiável.

- o bytecode checker analisa o bytecode presentado à execução e assegura que as regras básicas da gramática JAVA são obedecidas
- o class launcher entrega os tipos de dados necessários ao interpretador JAVA
- o security manager lida com os problemas que colocam potencialmente em risco o sistema de segurança relativo à aplicação, controlando as condições de acesso do programa que está a correr para o sistema de



ficheiros, para a network, para processos externos e para o sistema window.

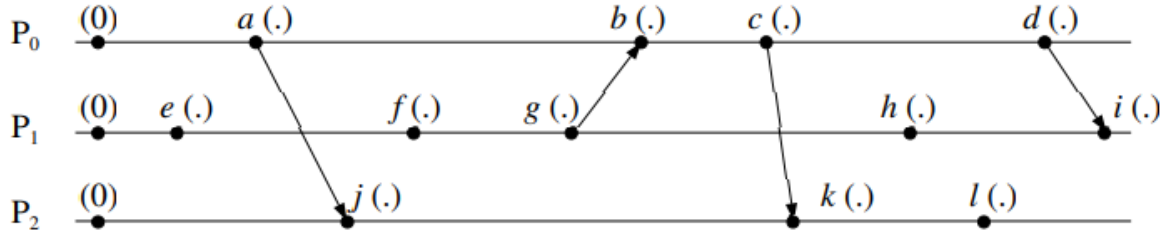
2 - Considere a troca de mensagens como o paradigma de comunicação que foi escolhido para se estabelecer um modelo cliente-servidor na variante de cópia de recursos entre processos residentes nos diferentes nós de processamento de um sistema distribuído e admita que existem duas cópias. Apresenta um diagrama esquemático que descreve a interação funcional entre os diferentes componentes de um tal sistema no lado do servidor, identificando cada um dos componentes introduzido e elucidando o seu papel. Explique que vantagens apresenta esta variante e em que consiste o problema da consistência da informação.



Variante tipo 3(Replicação de recursos) - o serviço é disponibilizado simultaneamente em múltiplos sistemas, cada um executando a variante do tipo 2(na qual os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo.), então o recurso partilhado é replicado em cada servidor o que resulta em múltiplas cópias. Este é um modelo sofisticado que tem como objetivo maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga(potencializando assim a escalabilidade)

- o serviço é mantido operacional contra a falha de servidores específicos
- as solicitações do cliente são distribuídas entre os servidores disponíveis usando uma política, pré-fornecido pelo serviço DNS, de associação geográfica para solicitações globais e de associação rotativa para solicitações locais
- quando há uma alteração de dados locais em uma das réplicas do recurso compartilhado, a necessidade de manter as diferentes réplicas consistentes surge e deve ser tratado.

3 - A figura abaixo descreve a evolução temporal de três processos cujos relógios locais são relógios lógicos escalares sincronizados segundo o algoritmo de acerto de Lamport.



a) Atribua aos diferentes acontecimentos, especificados por letras do alfabeto minúsculo (a ... l), que ocorrem nos três processos, o valor da marca temporal (time stamp) que lhes está associado.

b) Indique para os seguintes pares de acontecimentos, a-f, f-h, b-i, l-i, j-i e a-l, se se trata de acontecimentos concorrentes (||) ou de acontecimentos ligados por um nexo de causalidade (->).

a) a(1) b(4) c(5) d(6)
e(1) f(2) g(3) h(4) i(7)
j(2) k(6) l(7)

b) concorrentes: a||f l||i j||i
casualidade: f ->h b->i a->l

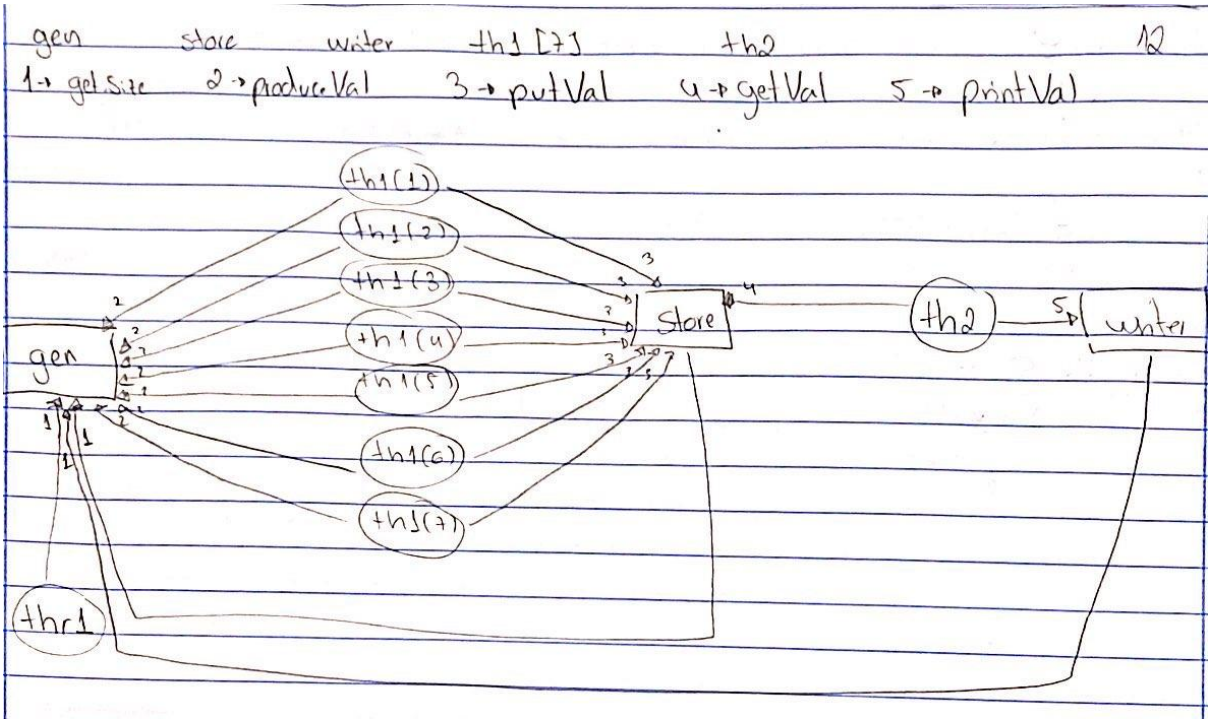
4 - Considere uma situação no contexto da comunicação entre pares em que coexistem três processos, P1, P2 e P3, que pretendem eleger um deles como chefe. Descreva os passos principais de um algoritmo que conduz a essa eleição. Admita que não há perda de mensagens, nem que qualquer dos processos falha. Como alteraria o algoritmo se as suposições anteriores deixassem de ser válidas.

Assumindo-se que não existem perdas de mensagens, cada um dos nós P1,P2 e P3 no início geram um número de identificação para si mesmo, e cada nó assume-se a si mesmo como líder, marca a mensagem como sua e passa-a ao nó seguinte, em sentido do relógio.
Após a receção da mensagem o nó seguinte, verifica o número de id. do líder e verifica se o seu número de identificação é maior, se for maior, substitui o número de identificação do líder na mensagem pelo seu, se não for maior não troca e reenvia a mensagem para o próximo nó.
As mensagens passam por todos os nós, até que as mensagens cheguem aos seus nós iniciais. Aqui cada um dos nós verifica a sua mensagem e vê se

o número de identificação do líder da sua mensagem , caso seja igual, esse nó identifica-se como líder, se não for igual, identifica-se como sendo um coordenador.

Parte B

1 - Representando as entidades activas por círculos e as entidades passivas por retângulos, faça um diagrama ilustrativo da interação em presença e indique por palavras simples qual é o papel desempenhado pelos threads de cada tipo (não mais do que uma frase).



Existem 7 threads do tipo 1 que vão buscar um valor ao genRegion(produceVal) e vão guardar o valor com o id da thread na storeRegion (putVal). A thread do tipo 2 vai buscar os valores guardados na store (getVal) e vai imprimir esses valores (printVal).

2017 - 14 junho

Parte A

1 - Tanenbaum defines a distributed system as a collection of independent computers that appears to its users as a single coherent system. Using this definition as the starting point, try to elicit some of the distinctive features that this kind of systems present, namely communication through message passing, failure handling and global internal state.

A definição tradicional de sistemas distribuídos afirma que se está a lidar com um sistema operacional cujo os componentes, localizado em diferentes nós de processamento de um sistema de computador paralelo, comunicam e coordenam as suas ações através de message passing.
Uma definição alternativa de sistemas distribuídos dada por Tanenbaum, embora essencialmente equivalente, é dizer que se trata de uma coleção de sistemas computacionais independentes percebidos pelos utilizadores como um sistema coerente.
Criando assim algumas características distintas que este sistema apresenta:

- Comunicação através de message passing (*Communication through message passing)* - o facto de nenhuma suposição ser feita de como os sistemas computacionais estão interligados, envolve um mecanismo minimalista baseado na troca de mensagens.
- Tratamento de falhas (*Failure handling*) - pois qualquer um dos componentes do sistema distribuído pode falhar a qualquer momento, deixando os componentes restantes em operação.
- Estado interno global (*global internal state*) - a necessidade de fornecer uma imagem de trabalho coerente requer alguma forma de coordenação de atividades entre os diferentes nós de processamento.

(restantes que não pede nesta pergunta)

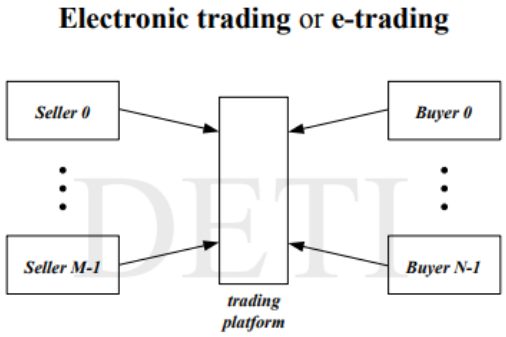
- paralelismo, pois a existência de sistemas computacionais independentes dá origem a threads autónomas de execução;

- escalabilidade, pois expandir o sistema através da integração de mais nós de processamento é um resultado imediato da organização prescrita;

2 - Describe the publisher-subscriber model and explain how it fits in two application areas where it has become very popular.

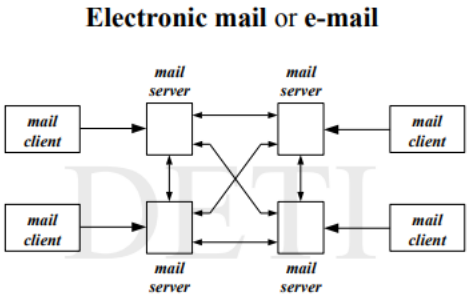
No modelo publisher-subscriber existem múltiplos fornecedores de serviço, os publishers, e múltiplos receptores de serviço, os subscribers, que estão totalmente dissociados uns dos outros por meio de mediação de um serviço intermediário, o broker. Umas das principais características deste modelo, em comparação com o modelo client-server convencional, é que não há nenhuma interação síncrona a funcionar ou seja as interações são feitas de forma assíncrona.
Duas áreas de aplicação que se tornou bastante popular são:

- e-commerce (e-trading) - nesta área os vendedores agem como publishers que adicionam a informação sobre os produtos que querem à venda na plataforma de negociação, enquanto os compradores agem como subscribers que pedem informação sobre os produtos a plataforma de negociação que têm intenções de comprar. A plataforma de negociação funciona como um broker que gere as possíveis transações entre vendedores e compradores.

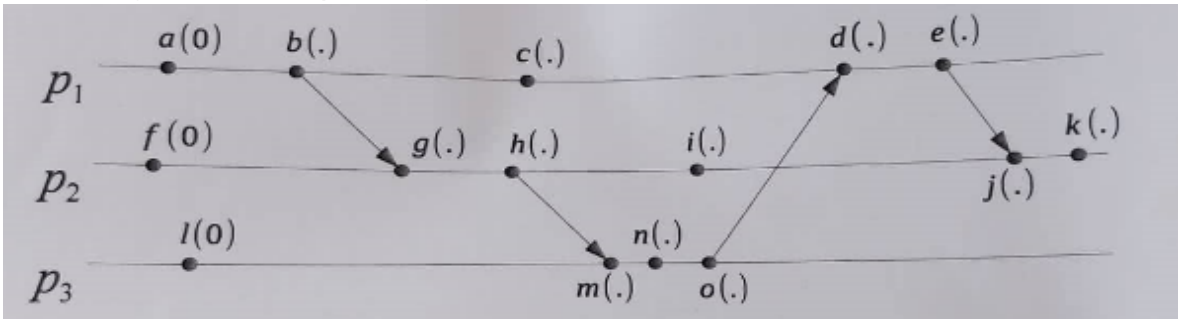


- e-mail - os servidores de email funcionam como brokers para os clientes de mail locais e para outros servidores de mail, para controlar as mensagens locais e também funcionam como publishers para outros

servidores de mail, para encaminhamento de mensagens locais. Os clientes de mail agem como publishers para mandar mensagens e como subscribers para receber mensagens.



3 - *The schematics below describes the temporal evolution of three processes whose local clocks are scalar logical clocks synchronized according to the Lamport algorithm.*

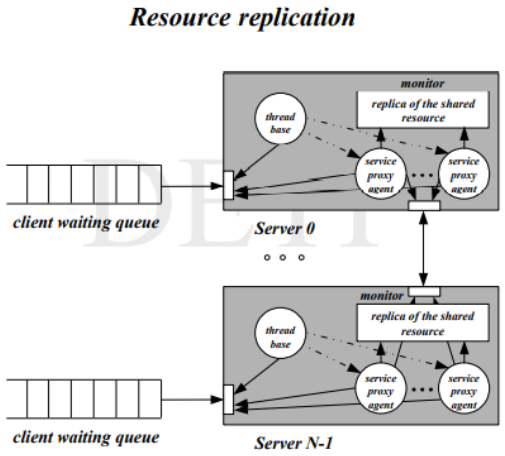


a) *Assign to the different events, specified by small letters(a..o), their associated timestamp.*
b) *State for the following event pairs, a-j, f-c, b-i, e-m, i-o and i-e, if they are concurrent (||) or sequential ones (->).*

a) a(0) b(1) c(2) d(7) e(8)
f(0) g(2) h(3) i(4) j(9) k(10)
l(0) m(4) n(5) o(6)

b) concurrent: i||o, i||e
sequential: a-> j, f ->c, b->i, e->m,

4 - Suppose we want to ensure service availability in a client-server model. In order to achieve this aim, a resource replication variante where the server is installed in two hardware platforms, is implemented. In principle, only one of the servers is active at a time and interacts with the clients. When it fails, the other start operating immediately, replacing it.
Draw a functional diagram that depicts the organization and list three problems which must be solved for the system to work properly. Justify clearly your claims



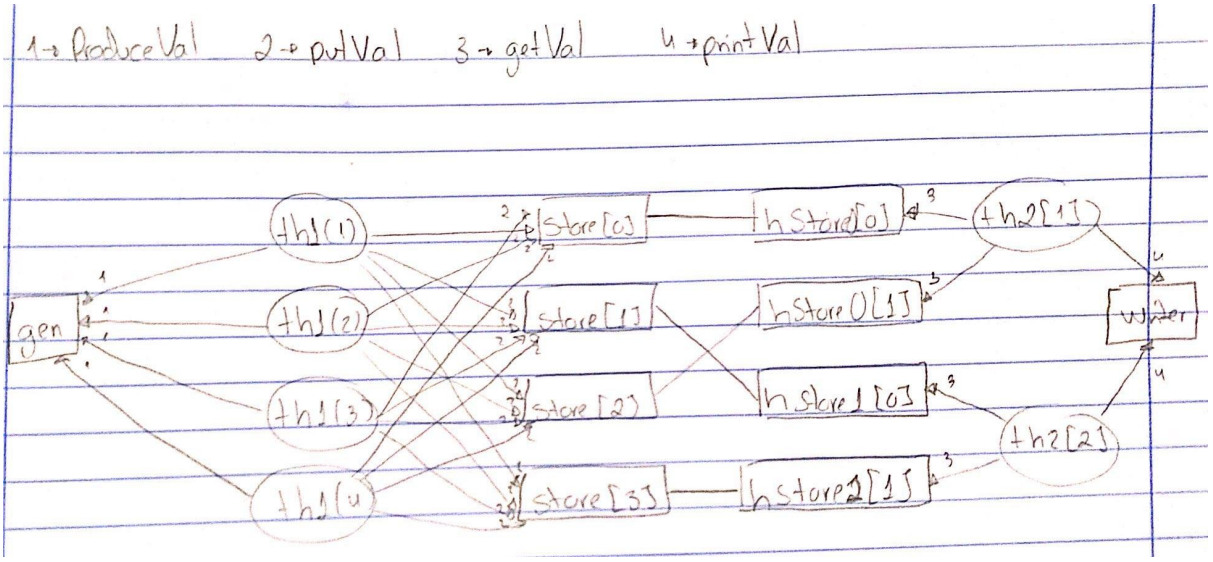
Variante tipo 3(Replicação de recursos) - o serviço é disponibilizado simultaneamente em múltiplos sistemas, cada um executando a variante do tipo 2(na qual os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo.), então o recurso partilhado é replicado em cada servidor o que resulta em múltiplas cópias. Este é um modelo sofisticado que tem como objetivo maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga(potencializando assim a escalabilidade). O serviço é mantido operacional contra a falha de servidores específicos. as solicitações do cliente são distribuídas entre os servidores disponíveis usando uma política, pré-fornecido pelo serviço DNS, de associação geográfica para solicitações globais e de associação rotativa para solicitações locais

- (consistência de informação) Quando há uma alteração de dados locais em uma das réplicas do recurso compartilhado, a necessidade de manter as diferentes réplicas consistentes surge e deve ser tratada dado isto, os servidores podem ser implementados em sistemas diferentes e neste caso é necessário assegurar que os sistemas tenham algumas semelhanças, nomeadamente a mesma capacidade a nível de memória.
- outro problema é que o serviço quando existe um pico de carga pode deixar de conseguir atender todos os processo, logo é necessário manter o serviço operacional contra a falha de servidores em particular

- 3º problema (dado que a exclusão mútua não é garantida nesta variante (e na variante 2) o recurso partilhado é transformado num monitor)

Parte B

1 - Representing the active entities by circles and the passive entities by squares, sketch a diagram which illustrates the interaction that is taking place and describe in simple words the role performed by the threads of each type.

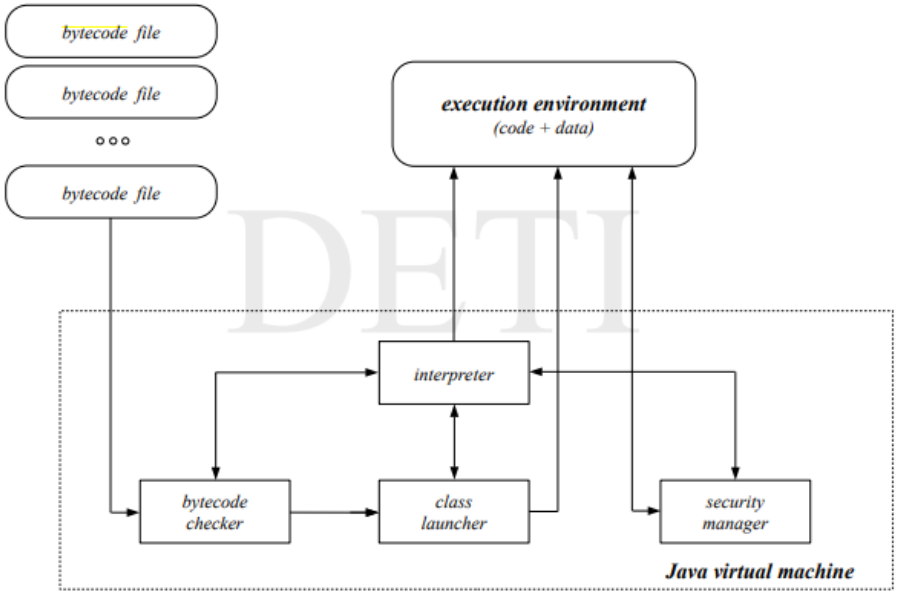


As quatro threads de tipo 1 vão buscar um valor ao genRegion (produceVal) e vão guardando, dependendo do valor que vai buscar, nas store[0 ... 3]. Como a store[0] = hstore0[0], store[1] = hstore1[0], store[2] = hstore0[1] e store[3] = hstore1[1], as 2 threads do tipo 2 vão buscar esses valores guardados , a thread2[1] aos hstore0 , e a thread2[2] as hstore1. Após terem os valores vão imprimi-los.

2018 normal - 20 Junho

Parte A

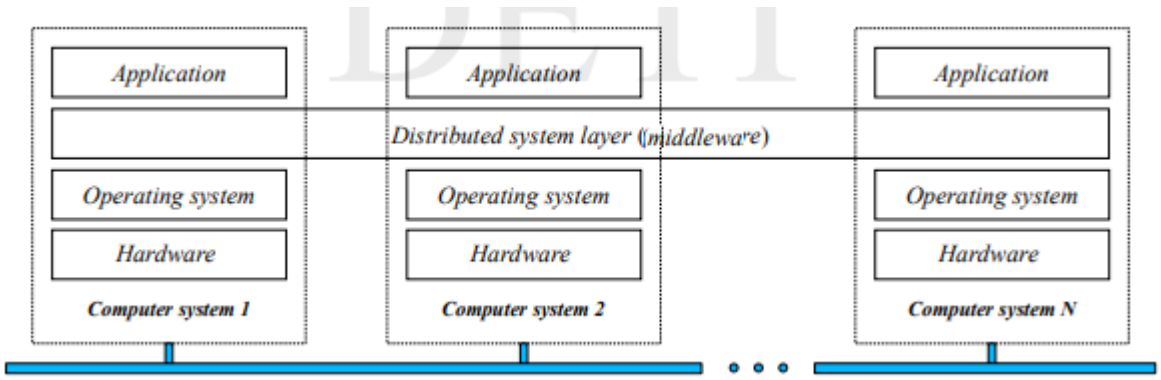
1 - The diagram below describes components of the Java Virtual Machine (JVM).



Define middleware and explain why the Java Virtual Machine (JVM) can be considered part of the middleware. Present clearly your claims.

Middleware pode ser descrito como um conjunto de processos que se comunicam entre si de forma a mascarar as heterogeneidades dos sistemas operativos de tal modo a que a aplicação possa ser executada em todos eles. A máquina virtual de Java (JVM) é constituída por uma camada chamada middleware que faz com que as aplicações sejam totalmente independentes da plataforma hardware e do Sistema operativo, implementado uma tripla camada de segurança que protege o sistema contra código não-confiável.

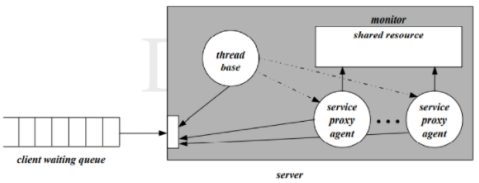
- o bytecode checker analisa o bytecode presentado à execução e assegura que as regras básicas da gramática JAVA são obedecidas
- o class launcher entrega os tipos de dados necessários ao interpretador JAVA



- o security manager lida com os problemas que colocam potencialmente em risco o sistema de segurança relativo à aplicação, controlando as condições de acesso do programa que está a correr para o sistema de ficheiros, para a network, para processos externos e para o sistema window.

2 - Take message passing as the communication paradigm which was chosen to establish a client-server model among processes residing in different nodes of a distributed system. Draw a diagram that describes the functional interaction among the components of such model, both at client and at the server side, identifying each one of them and explaining their role in the interaction. Assume the server replication variant.
Refer in this context to what one means by message marshaling and unmarshaling. How they are enforced in java

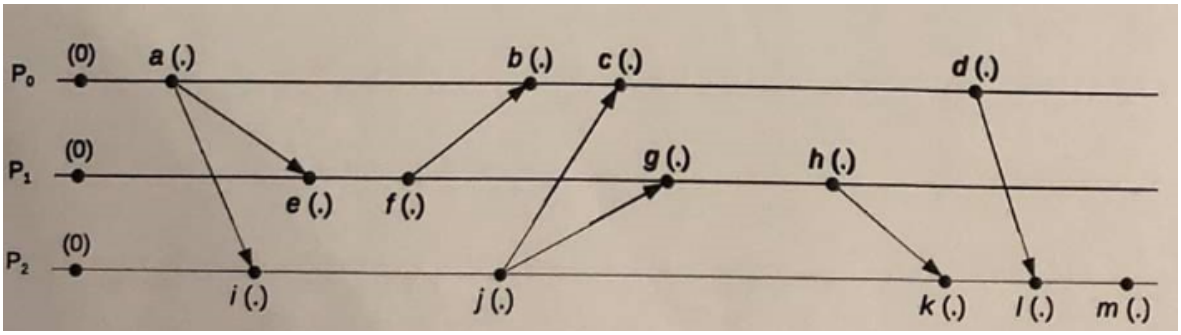
Variante do tipo 2(Replicação do servidor) - os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo. Esta é a forma tradicional de como os servidores são configurados, de forma a que se aproveite ao máximo os recursos do sistema do computador onde o servidor está localizado ao:



- o tempo de atendimento é minimizado, pois se aproveita a interação de tempos mortos através da simultaneidade
- permite a sincronização de diferentes processos do cliente no mesmo recurso compartilhado.

Marshaling e unmarshalling de informação é uma forma de garantir que a mensagem transmitida pelo canal de comunicação é segura, é vista como uma matriz de bytes que inclui os parâmetros, o seu tipo e como são estruturados, a construção de uma mensagem destas é chamada de marshaling e a operação de decodificação desta mensagem é chamada de unmarshalling.

3 - The schematics below depicts the temporal evolution of three processes whose local clocks are scalar logical clocks synchronized according to the Lamport algorithm.



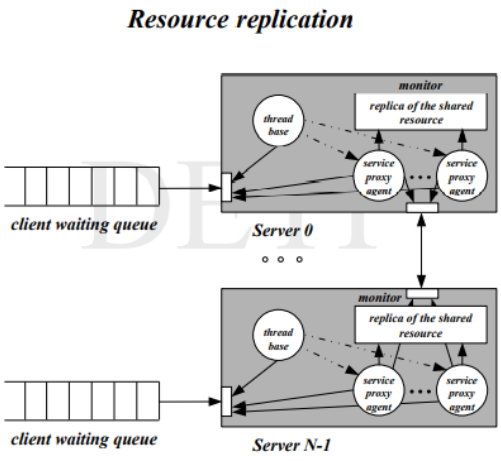
i. Assign to the different events, specified by small letters (a...m), their associated time stamp.
ii. Take from the schematics three examples of pairs of concurrent events (||) and list the longest sequence of sequential ones (->).
iii. Can the interaction depicted in the schematics be used as an example of access to a critical region with mutual exclusion in a distributed way? Present clearly your claims.

i. a(1) b(4) c(5) d(6)
e(2) f(3) g(4) h(5)
i(2) j(3) k(6) l(7) m(8)

ii. c||h j||b h||d a->e->f->b->d->l

iii. O diagrama de interação pode demonstrar acesso à região partilhada com exclusão mútua pois existem eventos concorrentes que acedem ao mesmo processo, como é o caso de f e j. Como são concorrentes, apenas um de cada vez pode aceder ao recurso (exclusão mútua)

4 - Suppose we want to ensure load balancing of the server tasks in a client-server model. In order to achieve this aim, a resource replication variant where the server is installed in two different platforms, is implemented. Both server instantiations are active at the same time and interact with the clients. Draw a functional diagram that depicts the organization and list three problems which must be solved for the system to work properly . Justify clearly your claims.



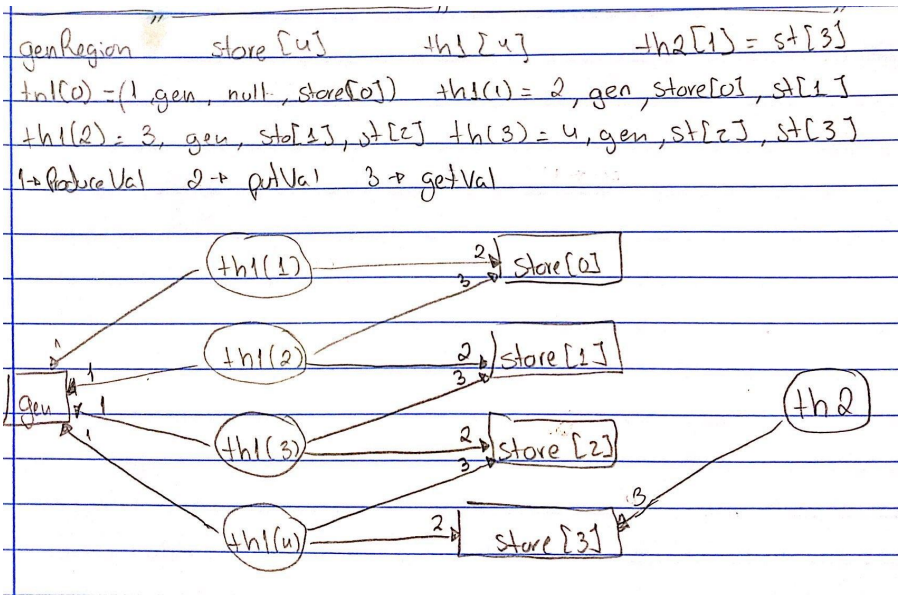
Variante tipo 3(Replicação de recursos) - o serviço é disponibilizado simultaneamente em múltiplos sistemas, cada um executando a variante do tipo 2(na qual os processos do cliente são

atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo.), então o recurso partilhado é replicado em cada servidor o que resulta em múltiplas cópias. Este é um modelo sofisticado que tem como objetivo maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga(potencializando assim a escalabilidade). O serviço é mantido operacional contra a falha de servidores específicos. as solicitações do cliente são distribuídas entre os servidores disponíveis usando uma política, pré-fornecido pelo serviço DNS, de associação geográfica para solicitações globais e de associação rotativa para solicitações locais

- (consistência de informação) Quando há uma alteração de dados locais em uma das réplicas do recurso compartilhado, a necessidade de manter as diferentes réplicas consistentes surge e deve ser tratada dado isto, os servidores podem ser implementados em sistemas diferentes e neste caso é necessário assegurar que os sistemas tenham algumas semelhanças, nomeadamente a mesma capacidade a nível de memória.
- outro problema é que o serviço quando existe um pico de carga pode deixar de conseguir atender todos os processo, logo é necessário manter o serviço operacional contra a falha de servidores em particular
- 3º problema (dado que a exclusão mútua não é garantida nesta variante (e na variante 2) o recurso partilhado é transformado num monitor)

Parte B

1 - Representing the active entities by circles and the passive entities by squares, sketch a diagram which illustrates the interaction that is taking place and describe in simple words the role performed by the threads of each type.



A thread do tipo 1 com o id 1 vais buscar um valor ao genRegion e guarda-o em store[0], as seguintes threads do tipo 1(2..4) vão também buscar um valor ao genRegion e vão buscar o valor ao store[i-2] (sendo i o seu id) e somar ao que foram buscar ao genRegion e guardar este valor no store[i-1]. A thread do tipo 2 vai buscar o resultado destas somas ao store[3] e imprime os resultados.

2 - Write an output of a single run. Bear in mind that, because of the concurrency and the randomness which were introduced, there is not an unique printing.

3 - The application is deadlock free.Explain why it so. Justify clearly your reasoning.

Dado que a thread do tipo 1 quando já não tem mais valores para ir buscar ao genRegion, recebe um 0 e guarda-o nas store que faz com que ao receber esses valores 0 faz com que as threads tipo 1 acabem os seu ciclo e então as threads ao receber um valor 0 da store[3] faz com que acabe também o seu ciclo, chegando assim ao fim do processo sem possibilidade de deadlock.

4 - Change the program so that the threads of type 2 process three values, instead of a single value, at each state of the iterative cycle as it happens now.Start by pointing out the changes that have to be made in the code. They should be minimal.

2018 recurso - 9 julho

Parte A

1 - What is meant by transparency in the context of distributed systems? Give examples, at least five, of the modes transparency may assume. Justify clearly your claims.

A transparência no contexto dos sistemas distribuídos é a capacidade deste se mostrar com um único computador, ou seja, é uma característica que demonstra, com maior ou menor sucesso, a forma de mascarar a complexidade da camada interior de um sistema, isto é, mascarar o facto de que os vários recursos e processos estão distribuídos em múltiplos computadores. Estas transparências podem assumir vários modos, como por exemplo:

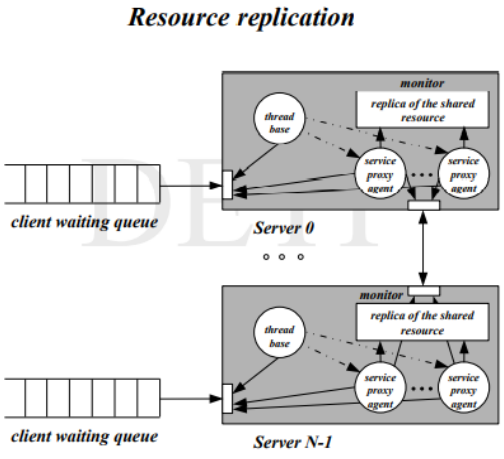
- transparência de acesso - quando as mesmas operações são feitas para aceder recursos locais e remotos
- transparência de posição - quando um acesso de um recurso é levado sem o conhecimento da sua localização física ou de rede
- transparência de rede - quando transparência de acesso e posição existem ao mesmo tempo
- transparência de movimento - quando a localização do acesso do cliente aos recursos pode mudar dentro do sistema sem afetar a operação a ser feita
- transparência de migração -quando os recursos podem ser movidos sem afetar o seu acesso
- transparência de relocalização- quando os recursos pode ser movidos sem afetar o seu acesso, mesmo quando um acesso está a ocorrer (versão mais forte de transparência de migração)
- transparência de replicação - quando é possível instanciar várias cópias do mesmo recurso sem que isso se torne óbvio
- transparência de desempenho - quando uma reconfiguração dinâmica do sistema pode ocorrer para lidar com as variações de carga
- transparência de escala - quando o sistema e os aplicativos podem expandir em escala sem exigir qualquer alteração na estrutura do sistema e no algoritmo da aplicação
- transparência de simultaneidade - quando o acesso a recursos compartilhados é realizado em paralelo por várias entidades sem

estarem cientes umas das outras (os dados devem permanecer sempre consistente)

- transparência de falha - quando falhas, ocorrendo em componentes de hardware e / ou software do sistema, podem ser mascarados e, portanto, as tarefas em execução podem ser encerradas.

2 - Take access to remote objects as the communication paradigm which was chosen to establish a client-server model among processes residing in different nodes of a distributed system. Draw a diagram that describes the distribution of the components of such a model, at the server side, by different computer systems, identifying each of them and explain their role in the interaction. Assume that there are 3 remote objects positioned in geographically distant regions.
Refer in this context to what one means by network transparency. How can it be enforced in Java ?

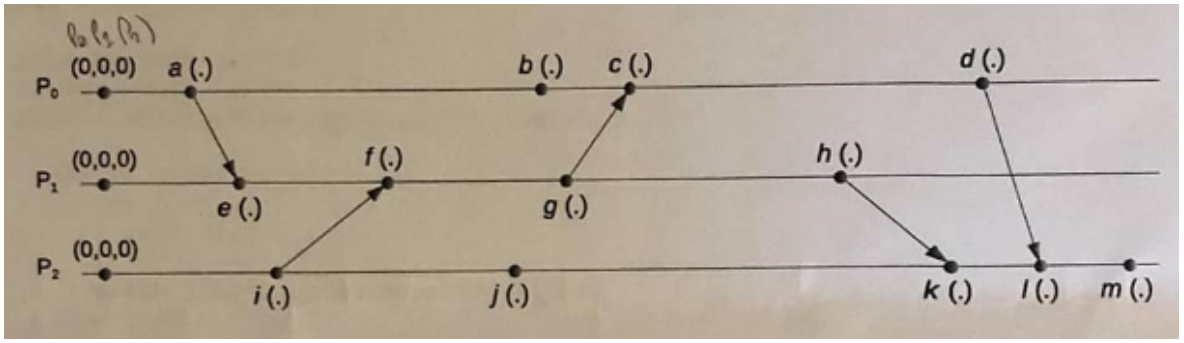
Como é dito no enunciado que estamos perante 3 objetos posicionados geograficamente em regiões distintas, estamos a falar da variante 3 do modelo client-server, que é a variante de réplica de recursos.



<u>Variante tipo 3(Replicação de recursos)</u> - o serviço é disponibilizado simultaneamente em múltiplos sistemas, cada um executando a variante do tipo 2(na qual os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo.), então o recurso partilhado é replicado em cada servidor o que resulta em múltiplas cópias. Este é um modelo sofisticado que tem como objetivo maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga(potencializando assim a escalabilidade). O serviço é mantido operacional contra a falha de servidores específicos, as solicitações do cliente são distribuídas entre os servidores disponíveis usando

uma política, pré-fornecido pelo serviço DNS, de associação geográfica para solicitações globais e de associação rotativa para solicitações locais. Quanto à transparência de rede, acontece quando a transparência de posição e de acesso acontecem ao mesmo tempo. Ou seja, transparência de posição é quando um acesso de um recurso é levado sem o conhecimento da sua localização física ou de rede e a transparência de acesso é quando as mesmas operações são feitas para aceder a recursos locais e remotos. O que neste tipo de variante acontece pois , o recurso partilhado é replicado em cada servidor, o que dá origem a múltiplas cópias, sendo isto uma transparência de acesso, enquanto as solicitações dos clientes vai ser distribuídas pelos pedidos globais (política de associação geográfica) e pelos pedidos locais(política de associação rotativa), o que é faz com que não se saiba a localização exata dos servidores o que é uma transparência de posição.

3 - The schematics below depicts the temporal evolution of three processes whose local clocks are vector logical clocks synchronized according to the vector algorithm.



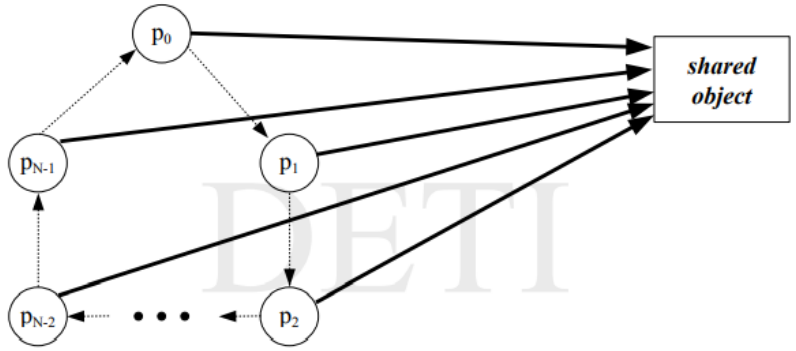
i. Assign to the different events, specified by small letters (a...m), their associated time stamp.
ii. Take from the schematics three examples of pairs of concurrent events (||) and list the longest sequence of sequential ones (->).
iii. Identify all the events that belong to the past of event g? Consider next all the events that belong to its future. Justify your reasoning clearly.

i. a(1,0,0) b(2,0,0) c(3,3,1) d(4,3,1)
e(1,1,0) f(1,2,1) g(1,3,1) h(1,4,1)
i(0,0,1) j(0,0,2) k(1,4,3) l(4,3,4)
m(4,3,5)

ii. i||e j||b c||h
a->e->f->g->c->d->l->m

iii. passado G : F I E A
futuro G: C D L H K

4 - In peer-to-peer communication, a virtual ring based on the individual id was established for communication through message passing among the active processing nodes of a distributed system. Describe in detail the algorithms which allow the insertion and the retrieval of a processing node in the ring. Draw a diagram for each case to help clarify your description. Assume that there are neither message loss, nor node crashing.

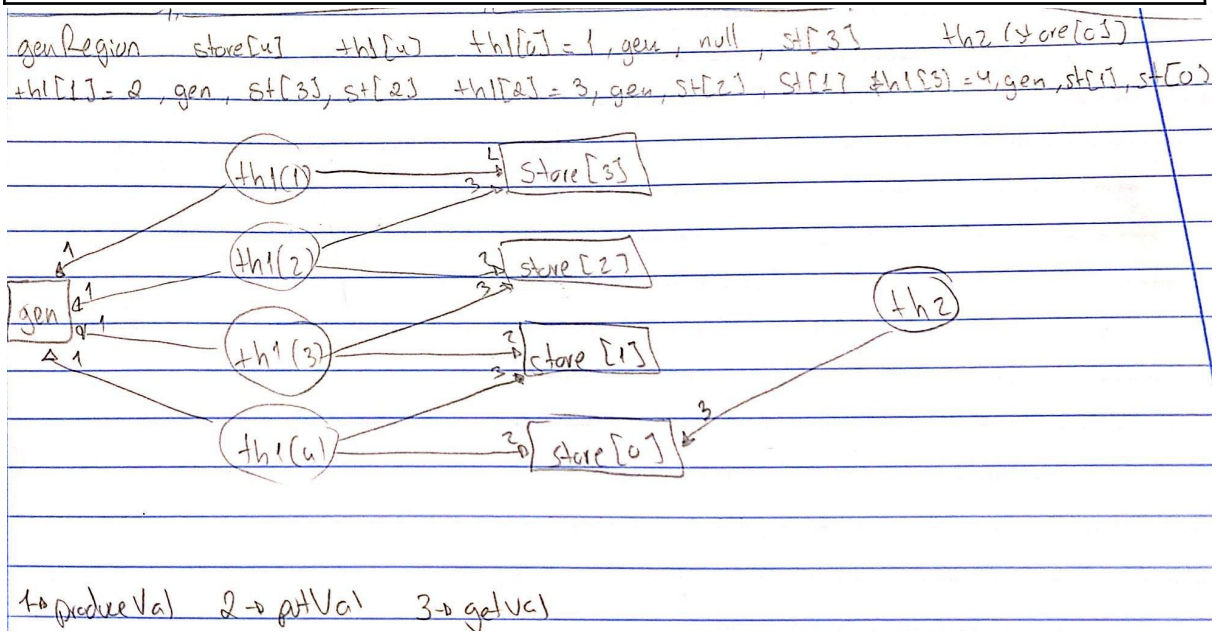


Um anel virtual é composto por um conjunto de processos que estão num ciclo fechado. A comunicação entre um processo e a área partilhada é feita unicamente pelo processo que tem naquele momento o ring token, que circula entre eles, ou seja, se um processo tiver intenções de aceder a area partilhada tem de esperar até receber o ring token.
Dado que não existem falhas, um dos processos do ring inicia a eleição e gera um número de identificação (id) para si mesmo e atribui esse id a mensagem e passa ao nó seguinte (sentido dos ponteiros do relógio). O nó seguinte, muda o seu estado para participante , já com o seu id gerado, compara o seu id com o da mensagem e se o seu id for maior que o da mensagem, substitui o id da mensagem pelo seu, se não for assume-se como participante, e manda novamente a mensagem para o próximo nó. E assim sucessivamente até que um processo receba uma mensagem que tenha o seu id , então esse processo é eleito líder, então envia uma mensagem do tipo elected com a sua própria identificação, cada processo ao receber essa mensagem , muda o seu estado para não-participante, e se o id for diferente do seu , guarda o id do líder e manda a mensagem para o próximo processo.

Quando o líder recebe esta mensagem, e o id da mensagem for igual a sua, descarta a mensagem que acaba com o processo de eleição.

Parte B

1 - Representing the active entities by circles and the passive entities by squares, sketch a diagram which illustrates the interaction that is taking place and describe in simple words the role performed by the threads of each type.

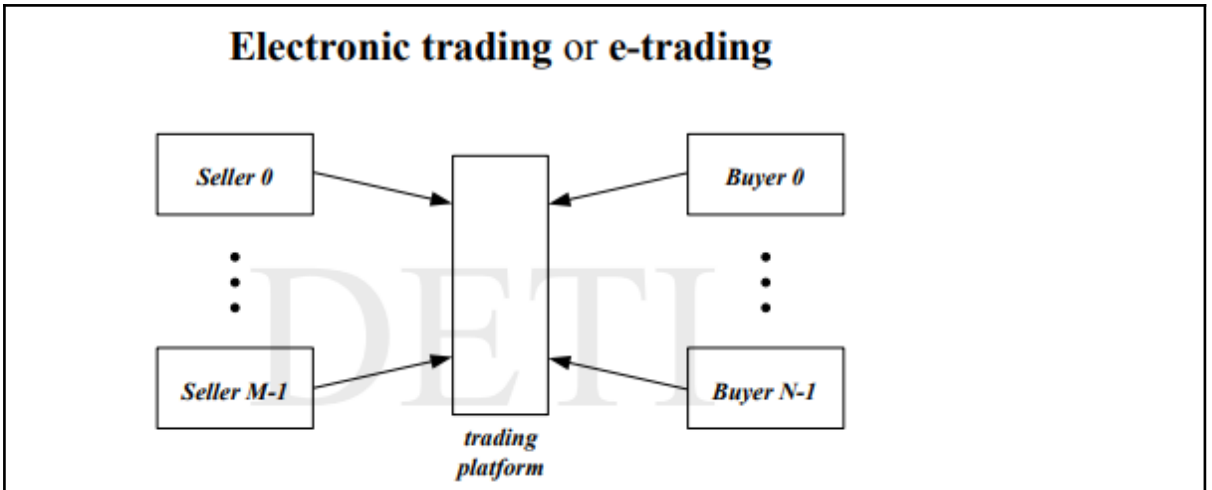


A thread do tipo 1 com o id 1 vais buscar um valor ao genRegion e guarda-o em store[3], as seguintes threads do tipo 1(2,3,4) vão também buscar um valor ao genRegion e vão buscar o valor ao store[3,2,1] (pela ordem das threads) e somar ao que foram buscar ao genRegion e guardar este valor no store[2,1,0]. A thread do tipo 2 vai buscar o resultado destas somas ao store[0] e imprime os resultados.

2019 normal - 12 junho

Parte A

1 - The diagram below describes the main components of the publisher-subscriber Model.

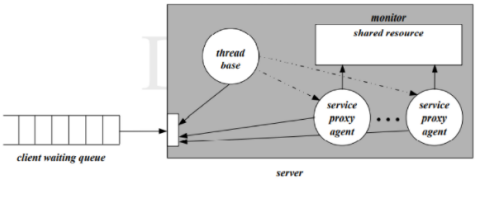


Describe how it operates in an e-commerce application. Are the interactions that take place synchronous or asynchronous? Present clearly your claims.

No modelo publisher-subscriber existem múltiplos fornecedores de serviço, os publishers, e múltiplos receptores de serviço, os subscribers, que estão totalmente dissociados uns dos outros por meio de mediação de um serviço intermediário, o broker. Umas das principais características deste modelo, em comparação com o modelo client-server convencional, é que não há nenhuma interação síncrona a funcionar ou seja as interações são feitas de forma assíncrona. Na aplicação de e-commerce (e-trading), os vendedores agem como publishers que adicionam a informação sobre os produtos que querem à venda na plataforma de negociação, enquanto os compradores agem como subscribers que pedem informação sobre os produtos a plataforma de negociação que têm intenções de comprar. A plataforma de negociação funciona como um broker que gere as possíveis transações entre vendedores e compradores.

2 - Take access to remote objects,implemented by JAVA RMI, as the communication paradigm which was selected to establish a client-server model among processes residing in a distributed system. Draw a diagram that describes the functionall interaction among the components of such a model,both at the client and at the server side, identifying each of them and explaining their role int the interaction.
Assume the server replication variant.
Refer in this context to how object transparency is implemented.

Variante do tipo 2(Replicação do servidor) - os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a

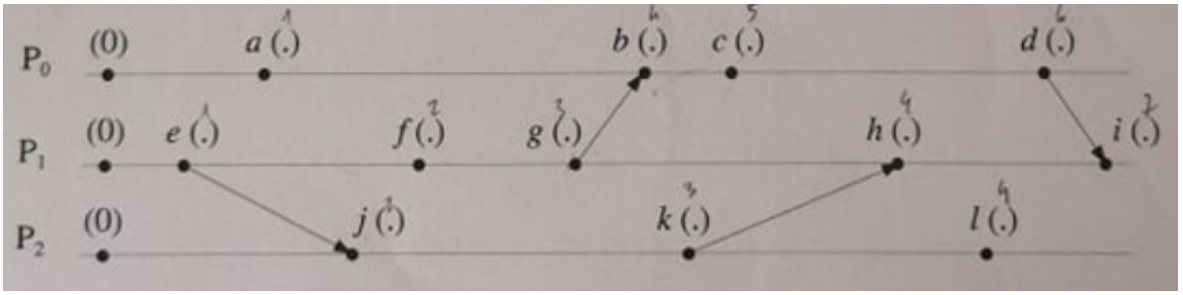


exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo. Esta é a forma tradicional de como os servidores são configurados, de forma a que se aproveite ao máximo os recursos do sistema do computador onde o servidor está localizado ao:

- o tempo de atendimento é minimizado, pois se aproveita a interação de tempos mortos através da simultaneidade
- permite a sincronização de diferentes processos do cliente no mesmo recurso compartilhado.

Com o RMI a transparência de objetos é conseguida uma vez que o nível de abstração permite invocar objetos remotos da mesma forma que se invocam objetos locais (toda esta lógica é escondida, simplificando o código). A transparência de rede é tratada pelo Java pois a JVM usa os seus processos para mascarar as heterogeneidades do sistema e trata da comunicação dos sistemas de forma transparente ao programador. Com isto concluímos que, como o acesso a recursos locais e remotos é feito da mesma forma, e que a localização precisa dos recursos não é conhecida pelo utilizador, a transparência de rede é estabelecida.

3 - The schematics below depicts the temporal evolution of three processes whose local clocks are scalar logical clocks synchronized according to the Lamport algorithm.



i. Assign to the different events, specified by small letters (a...l), their associated time stamp.
ii. Take from the schematics three examples of pairs of concurrent events (||) and list the longest sequence of sequential ones (->).
iii. Explain why, taking examples from schematics, the time stamp associated with each event can not be used to carry out the casual ordering of events. There is, however, a special case where this is possible. Which is it? Present clearly your claims.

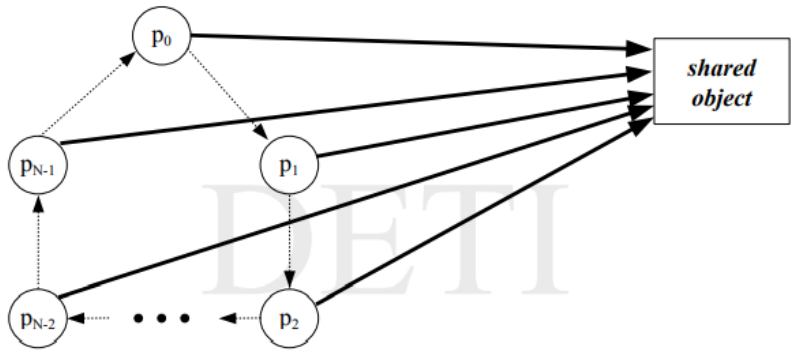
i. a(1) b(4) c(5) d(6)
e(1) f(2) g(3) h(4) i(7)

j(2) k(3) l(4)

ii. a||e j||f k||g

e->f->g->b->c->d->i

4 - In peer-to-peer communication, a virtual ring based on the individual id was established for communication through message passing among the active processing nodes of a distributed system. Describe in detail the algorithms which allow the insertion and the retrieval of a processing node in the ring. Draw a diagram for each case to help clarify your description. Assume that there are neither message loss, nor node crashing.



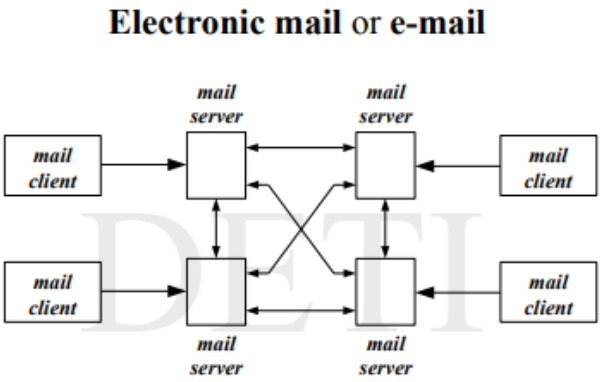
Um anel virtual é composto por um conjunto de processos que estão num ciclo fechado. A comunicação entre um processo e a área partilhada é feita unicamente pelo processo que tem naquele momento o ring token, que circula entre eles, ou seja, se um processo tiver intenções de aceder a area partilhada tem de esperar até receber o ring token.

Dado que não existem falhas, um dos processos do ring inicia a eleição e gera um número de identificação (id) para si mesmo e atribui esse id a mensagem e passa ao nó seguinte (sentido dos ponteiros do relógio). O nó seguinte, muda o seu estado para participante , já com o seu id gerado, compara o seu id com o da mensagem e se o seu id for maior que o da mensagem, substitui o id da mensagem pelo seu, se não for assume-se como participante, e manda novamente a mensagem para o próximo nó. E assim sucessivamente até que um processo receba uma mensagem que tenha o seu id , então esse processo é eleito líder, então envia uma mensagem do tipo elected com a sua própria identificação, cada processo ao receber essa mensagem , muda o seu estado para não-participante, e se o id for diferente do seu , guarda o id do líder e manda a mensagem para o próximo processo. Quando o líder recebe esta mensagem, e o id da mensagem for igual a sua, descarta a mensagem que acaba com o processo de eleição.

2020 normal - 30 junho

Parte A

1 - Eletronic mail, or e-mail, depicted in the schematics below, is a distributed application that follows the principles of the publisher-subscriber Model.



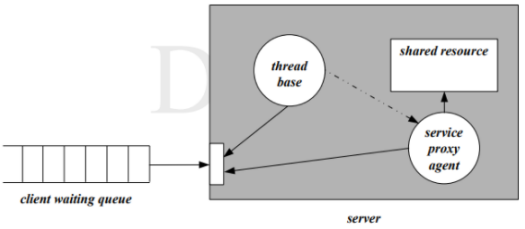
How does this model operates and what are its main components? Match the proprieties of these components to the aggregate of mail servers and clients that embody the mailing service. Present clearly your claims.

No modelo publisher-subscriber existem múltiplos fornecedores de serviço, os publishers, e múltiplos receptores de serviço, os subscribers, que estão totalmente dissociados uns dos outros por meio de mediação de um serviço intermediário, o broker. Umas das principais características deste modelo, em comparação com o modelo client-server convencional, é que não há nenhuma interação síncrona a funcionar ou seja as interações são feitas de forma assíncrona. na aplicação de e-mail os servidores de mail funcionam como brokers para os clientes de mail locais e para outros servidores de mail, para controlar as mensagens locais e também funcionam como publishers

para outros servidores de mail, para encaminhamento de mensagens locais. Os clientes de email agem como publishers para mandar mensagens e como subscribers para receber mensagens.

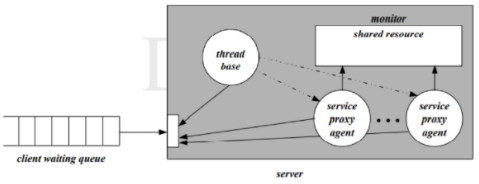
2 - take message passing as the communication paradigm which was chosen to establish a client-server model among processes residing in different nodes of a distributed system. Three variants of this model were studied. Describe them. Which of these is more commonly used and which is more reliable?
Present clearly your claims.
Refer in this context to what does one mean by message marshaling and unmarshaling. How are they enforced in Java?

Variante do tipo 1(solicitação de serialização) - apenas um processo do cliente é atendido de cada vez, o que significa que o thread base, após receber uma solicitação de conexão instância um agente proxy de serviço e espera que acabe(feche) antes de começar a ouvir novamente. Dado que apenas atente um processo de cada vez e que apenas um agente proxy de serviço está ativo, nenhuma segurança é necessária para garantir exclusão mútua na sessão de acesso. Mas dado isto, é uma variante bastante ineficiente pois :



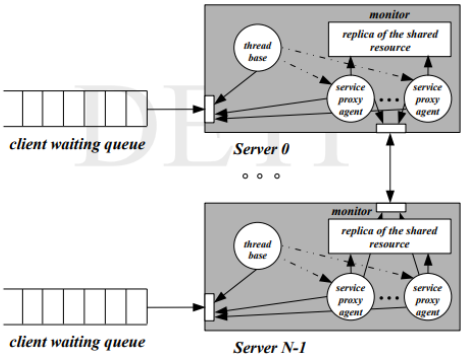
- o tempo de atendimento não é minimizado, dado que não aproveita os tempos mortos de interação , devido a falta de competição
- dá origem a uma espera ocupada na tentativa de sincronizar vários clientes que processam no mesmo recurso partilhado

Variante do tipo 2(Replicação do servidor) - os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo. Esta é a forma tradicional de como os servidores são configurados, de forma a que se aproveite ao máximo os recursos do sistema do computador onde o servidor está localizado ao:



- o tempo de atendimento é minimizado, pois se aproveita a interação de tempos mortos através da simultaneidade
- permite a sincronização de diferentes processos do cliente no mesmo recurso compartilhado.

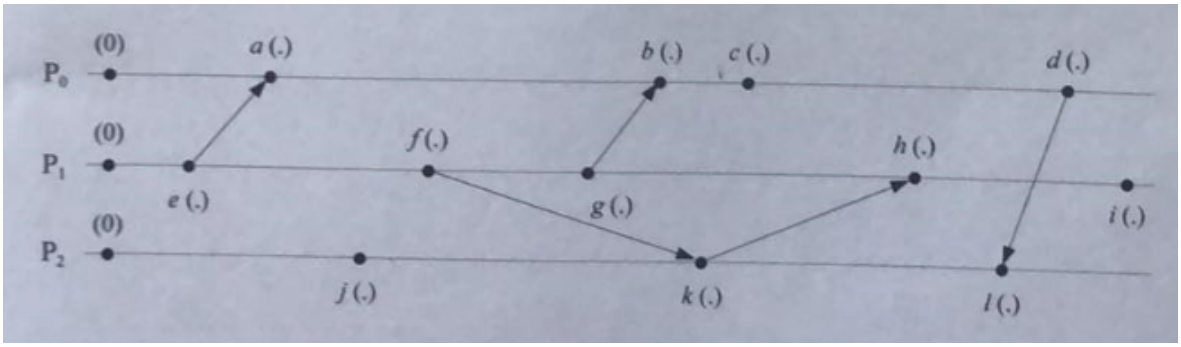
Variante tipo 3(Replicação de recursos) - o serviço é disponibilizado simultaneamente em múltiplos sistemas, cada um executando a variante do tipo 2(na qual os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo.), então o recurso partilhado é replicado em cada servidor o que resulta em múltiplas cópias. Este é um modelo sofisticado que tem como objetivo maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga(potencializando assim a escalabilidade)



- o serviço é mantido operacional contra a falha de servidores específicos
- as solicitações do cliente são distribuídas entre os servidores disponíveis usando uma política, pré-fornecido pelo serviço DNS, de associação geográfica para solicitações globais e de associação rotativa para solicitações locais
- quando há uma alteração de dados locais em uma das réplicas do recurso compartilhado, a necessidade de manter as diferentes réplicas consistentes surge e deve ser tratado.

Marshaling e unmarshalling de informação é uma forma de garantir que a mensagem transmitida pelo canal de comunicação é segura, é vista como uma matriz de bytes que inclui os parâmetros, o seu tipo e como são estruturados, a construção de uma mensagem destas é chamada de marshaling e a operação de decodificação desta mensagem é chamada de unmarshalling.

3 - The schematics below depicts the temporal evolution of three processes whose local clocks are scalar logical clocks synchronized according to the Lamport algorithm.



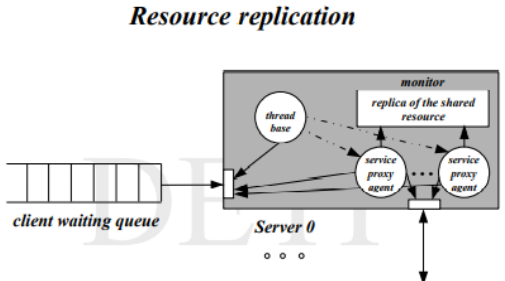
i. Assign to the different events, specified by small letters (a...l), their associated time stamp.
ii. Take from the schematics three examples of pairs of concurrent events (||) and list the longest sequence of sequential ones (->).
iii. Explain why, logical clocks, either of the scalar type or the vector type, are superior to ordinary time clocks in eliciting a casual ordering of events. Present clearly your claims.

i. a(2) b(4) c(5) d(6)
e(1) f(2) g(3) h(4) i(5)
j(1) k(3) l(7)

ii. j||a f||b c||k
e ->f->g->b->c->d->l

iii. Lamport sugeriu um dispositivo que ele chamou como relógio lógico que é essencialmente um contador local de eventos que não tem qualquer associação com hora real. O que faz com que seja bastante superior no que toca a um relógio de tempo normal , dado que o tempo não se pode medir diretamente (astronômico/ atômico) é calculado um padrão definido por uma média, que dependendo das condições, o que não é tão eficaz como um relógio lógico .

4 - Suppose one aims to improve service availability in a client-server model. In order to achieve this goal, a resource replication variant of the server is installed in two hardware platforms, located side by side. Only one of the units is active at a given time. However, if it fails, the other unit must immediately take its place in dealing with client requests.
Sketch a diagram describing the required organization and list three problems that must be solved so that the system works as expected. Present clearly your claims.



Variante tipo 3(Replicação de recursos) - o serviço é disponibilizado

simultaneamente em múltiplos sistemas, cada um executando a variante do tipo 2(na qual os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo.), então o recurso partilhado é replicado em cada servidor o que resulta em múltiplas cópias. Este é um modelo sofisticado que tem como objetivo maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga(potencializando assim a escalabilidade). O serviço é mantido operacional contra a falha de servidores específicos. as solicitações do cliente são distribuídas entre os servidores disponíveis usando uma política, pré-fornecido pelo serviço DNS, de associação geográfica para solicitações globais e de associação rotativa para solicitações locais

- (consistência de informação) Quando há uma alteração de dados locais em uma das réplicas do recurso compartilhado, a necessidade de manter as diferentes réplicas consistentes surge e deve ser tratada dado isto, os servidores podem ser implementados em sistemas diferentes e neste caso é necessário assegurar que os sistemas tenham algumas semelhanças, nomeadamente a mesma capacidade a nível de memória.
- outro problema é que o serviço quando existe um pico de carga pode deixar de conseguir atender todos os processo, logo é necessário manter o serviço operacional contra a falha de servidores em particular

2020 recurso - 21 julho

1 - What is meant by transparency in the context of distributed systems? Give examples, at least five, of the modes transparency may assume. Justify clearly your claims.

A transparência no contexto dos sistemas distribuídos é a capacidade deste se mostrar com um único computador, ou seja, é uma característica que demonstra, com maior ou menor sucesso, a forma de mascarar a complexidade da camada interior de um sistema, isto é, mascarar o facto de que os vários recursos e processos estão distribuídos em múltiplos

computadores. Estas transparências podem assumir vários modos, como por exemplo:

- transparência de acesso - quando as mesmas operações são feitas para aceder recursos locais e remotos
- transparência de posição - quando um acesso de um recurso é levado sem o conhecimento da sua localização física ou de rede
- transparência de rede - quando transparência de acesso e posição existem ao mesmo tempo
- transparência de movimento - quando a localização do acesso do cliente aos recursos pode mudar dentro do sistema sem afetar a operação a ser feita
- transparência de migração -quando os recursos podem ser movidos sem afetar o seu acesso
- transparência de relocalização- quando os recursos pode ser movidos sem afetar o seu acesso, mesmo quando um acesso está a ocorrer (versão mais forte de transparência de migração)
- transparência de replicação - quando é possível instanciar várias cópias do mesmo recurso sem que isso se torne óbvio
- transparência de desempenho - quando uma reconfiguração dinâmica do sistema pode ocorrer para lidar com as variações de carga
- transparência de escala - quando o sistema e os aplicativos podem expandir em escala sem exigir qualquer alteração na estrutura do sistema e no algoritmo da aplicação
- transparência de simultaneidade - quando o acesso a recursos compartilhados é realizado em paralelo por várias entidades sem estarem cientes umas das outras (os dados devem permanecer sempre consistente)
- transparência de falha - quando falhas, ocorrendo em componentes de hardware e / ou software do sistema, podem ser mascarados e, portanto, as tarefas em execução podem ser encerradas.

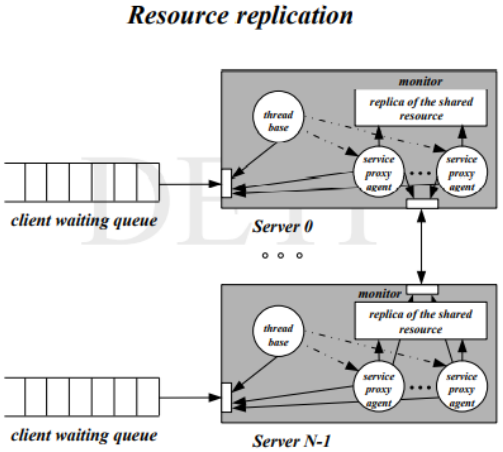
2 - take method invocation on remote objects as the communication paradigm which was chosen to establish a client-server model among processes residing in different nodes of a distributed system and assume that Java RMI was used to implement it.
Answer the following questions in a clear manner.
i. What is the main difference between invoking a method on a remote and a local object?
ii. Sketch a diagram describing the functional interaction of the different components of such a system at the server side

iii. When code migration for remote execution is being considered, it is essential to protect the object, and the hardware platform where it resides, against the execution of malicious code. Explain how this may be achieved.

i. O método de invocação remota(RMI) permite que haja interação de programas que estão a ser executados em diferentes nós de um sistema distribuído.(pelas RPC - Remote Producer Calls)
Com o RMI a transparência de objetos é conseguida uma vez que o nível de abstração permite invocar objetos remotos da mesma maneira que se invocam objetos locais.
existem 3 características a ter em atenção , pois uma RMI não funciona exatamente da mesma forma que a chamada de procedimento local:

- A chamada pode falhar, pois pode ainda não estar instalada, ou a infraestrutura de comunicação não estar a funcionar corretamente.
- Os parâmetros do procedimento e de valores de return devem ser passado por valor , pois como estão em diferentes endereços, o meio de comunicação entre eles é por passar dados relevantes juntamente com o seu formato e estrutura.
- e o procedimento de execução remota demora mais tempo que a execução de procedimentos local.

ii. dado que ha processos a residir em difernetes nos estamos perante a variante de replica de recursos

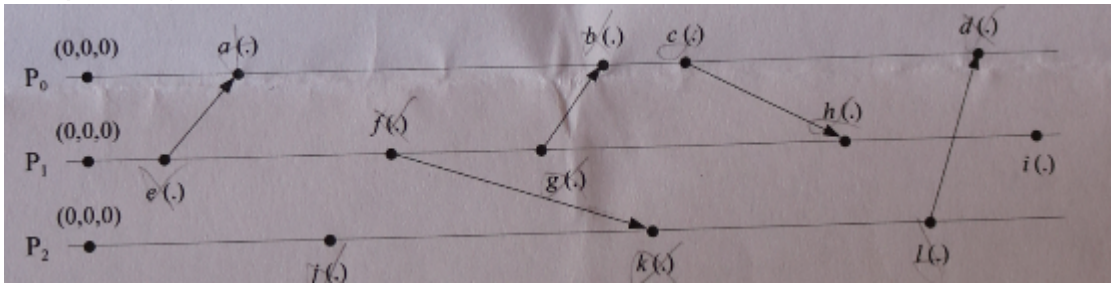


Variante tipo 3(Replicação de recursos) - o serviço é disponibilizado simultaneamente em múltiplos sistemas, cada um executando a variante do tipo 2(na qual os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo.), então o recurso partilhado é replicado em cada servidor o que resulta em múltiplas cópias. Este é um modelo sofisticado que tem como objetivo maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga(potencializando assim a escalabilidade). O serviço é mantido operacional contra a falha de servidores específicos. as solicitações do cliente são distribuídas entre os servidores

disponíveis usando uma política, pré-fornecido pelo serviço DNS, de associação geográfica para solicitações globais e de associação rotativa para solicitações locais

- (consistência de informação) Quando há uma alteração de dados locais em uma das réplicas do recurso compartilhado, a necessidade de manter as diferentes réplicas consistentes surge e deve ser tratada dado isto, os servidores podem ser implementados em sistemas diferentes e neste caso é necessário assegurar que os sistemas tenham algumas semelhanças, nomeadamente a mesma capacidade a nível de memória.
- outro problema é que o serviço quando existe um pico de carga pode deixar de conseguir atender todos os processo, logo é necessário manter o serviço operacional contra a falha de servidores em particular

3 - The schematics below depicts the temporal evolution of three processes whose local clocks are scalar logical clocks synchronized according to the Lamport algorithm.



i. Assign to the different events, specified by small letters (a...l), their associated time stamp.
ii. Take from the schematics three examples of pairs of concurrent events (||) and list the longest sequence of sequential ones (->).
iii. Explain why, logical clocks, either of the scalar type or the vector type, are superior to ordinary time clocks in eliciting a casual ordering of events. Present clearly your claims.

i. a(1,1,0) b(2,3,0) c(3,3,0) d(4,3,3)
e(0,1,0) f(0,2,0) g(0,3,0) h(3,4,0) i(3,5,0)
j(0,0,1) k(0,2,2) l(0,2,3)

ii. j||a k||b h||l
e->f->g->b->c->h->i

iii. Lamport sugeriu um dispositivo que ele chamou como relógio lógico que é essencialmente um contador local de eventos que não tem qualquer associação com hora real. O que faz com que seja bastante superior no que

toca a um relógio de tempo normal , dado que o tempo não se pode medir diretamente (astronômico/ atômico) é calculado um padrão definido por uma média, que dependendo das condições, o que não é tão eficaz como um relógio lógico .

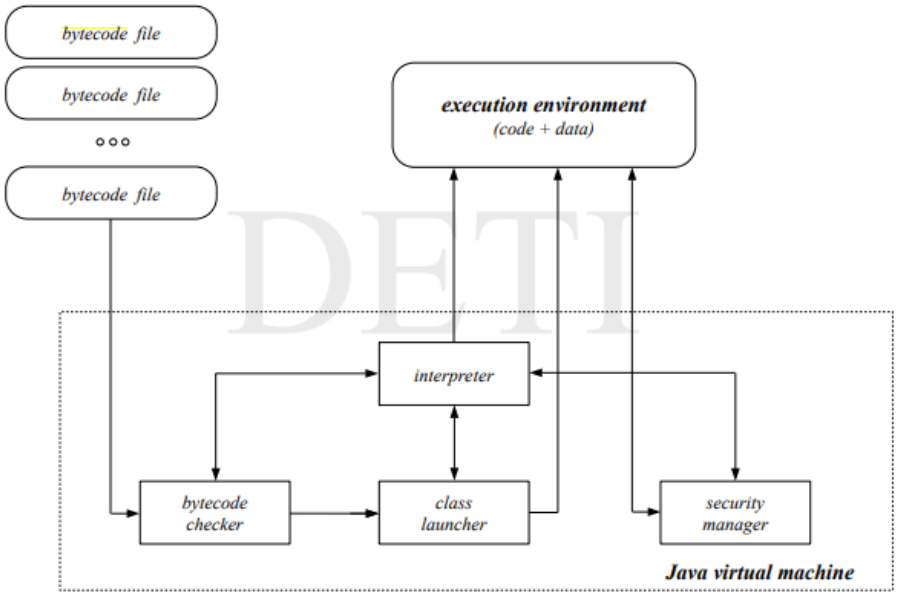
4 - Consider a situation in the context of peer-to-peer communication where three processes P1, P2 e P3 are required to elect a leader. Describe in detail the main steps of an algorithm that carries out the election. Assume there are no message loss and no process fails. How would you change the alghorithm if these hypothesis were no longer valid?

Dado que não existem falhas, um dos processos inicia a eleição e gera um número de identificação (id) para si mesmo e atribui esse id a mensagem e passa ao nó seguinte (sentido dos ponteiros do relógio). O nó seguinte, muda o seu estado para participante , já com o seu id gerado, compara o seu id com o da mensagem e se o seu id for maior que o da mensagem, substitui o id da mensagem pelo seu, se não for assume-se como participante, e manda novamente a mensagem para o próximo nó. E assim sucessivamente até que um processo receba uma mensagem que tenha o seu id , então esse processo é eleito líder, então envia uma mensagem do tipo elected com a sua própria identificação, cada processo ao receber essa mensagem , muda o seu estado para não-participante, e se o id for diferente do seu , guarda o id do líder e manda a mensagem para o próximo processo. Quando o líder recebe esta mensagem, e o id da mensagem for igual a sua, descarta a mensagem que acaba com o processo de eleição.

2020 - 28 Junho

Parte A

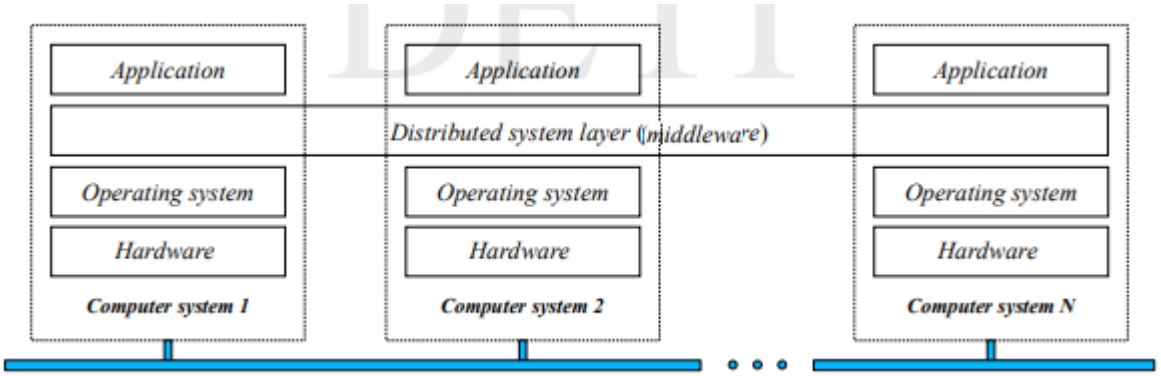
1 - The diagram below describes components of the Java Virtual Machine (JVM).



Define middleware and explain why the Java Virtual Machine (JVM) can be considered part of the middleware. Present clearly your claims.

Middleware pode ser descrito como um conjunto de processos que se comunicam entre si de forma a mascarar as heterogeneidades dos sistemas operativos de tal modo a que a aplicação possa ser executada em todos eles. A máquina virtual de Java (JVM) é constituída por uma camada chamada middleware que faz com que as aplicações sejam totalmente independentes da plataforma hardware e do Sistema operativo, implementado uma tripla camada de segurança que protege o sistema contra código não-confiável.

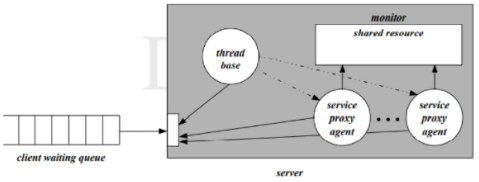
- o bytecode checker analisa o bytecode presentado à execução e assegura que as regras básicas da gramática JAVA são obedecidas
- o class launcher entrega os tipos de dados necessários ao interpretador JAVA
- o security manager lida com os problemas que colocam potencialmente em risco o sistema de segurança relativo à aplicação, controlando as condições de acesso do programa que está a correr para o sistema de



ficheiros, para a network, para processos externos e para o sistema window.

2 - Take message passing as the communication paradigm which was chosen to establish a client-server model among processes residing in different nodes of a distributed system. Draw a diagram that describes the functional interaction among the components of such model, both at client and at the server side, identifying each one of them and explaining their role in the interaction. Assume the server replication variant.
Refer in this context to what one means by message marshaling and unmarshaling. How they are enforced in java

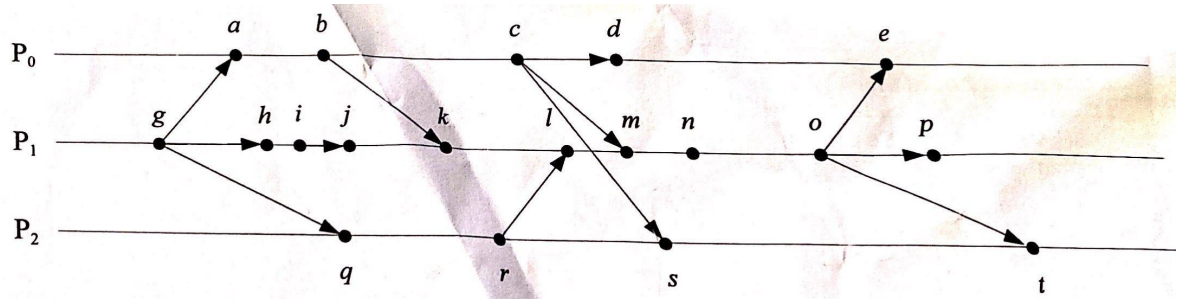
Variante do tipo 2(Replicação do servidor) - os processos do cliente são atendidos simultaneamente, ou seja, a thread base ao receber solicitação de conexão, instância um agente proxy de serviço e fica novamente a escuta de novas solicitações de conexão. Para garantir a exclusão mútua no acesso, o recurso compartilhado é transformado num monitor, dado que agora existem vários agentes proxy de serviço ativos ao mesmo tempo. Esta é a forma tradicional de como os servidores são configurados, de forma a que se aproveite ao máximo os recursos do sistema do computador onde o servidor está localizado ao:



- o tempo de atendimento é minimizado, pois se aproveita a interação de tempos mortos através da simultaneidade
- permite a sincronização de diferentes processos do cliente no mesmo recurso compartilhado.

Marshaling e unmarshalling de informação é uma forma de garantir que a mensagem transmitida pelo canal de comunicação é segura, é vista como uma matriz de bytes que inclui os parâmetros, o seu tipo e como são estruturados, a construção de uma mensagem destas é chamada de marshaling e a operação de decodificação desta mensagem é chamada de unmarshalling.

3 - clocks are temporal evolutiona of 3 processes whose local clocks synchronized according to the Lamport algorithm.



i. Assign to the different events, specified by small letters (a...t), their associated time stamp.
ii. Take from the schematics three examples of pairs of concurrent events (||) and list the longest sequence of sequential ones (->).
iii. Identify all the events that belong to the past of event l? Consider next all the events that belong to its future. Justify your reasoning clearly.

i. a(2) b(3) c(4) d(5) e(10)
g(1) h(2) i(3) j(4) k(5) l(6) m(7) n(8) o(9) p(10)
q(2) r(3) s(5) t(10)

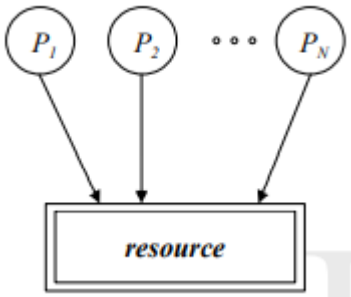
ii. i||b c||l e||p
g->h->i->j->k->l->m->n->o->t

iii. Passado de L: G H I J Q R K B A
Futuro de L: M N O P E T

4 - Assume a case of group communication where the processes exchange messages with one another and access a shared resource. Describe in detail an algorithm that allow them to access the resource in mutual exclusion in a distributed fashion. How is mutual exclusion effectively ensured ?
Draw diagrams to help clarify your description. Assume that there is neither message loss, nor process malfunction.

Num ambiente multiprogramado,os processos que existem nesse mesmo ambiente podem apresentar comportamentos diferenciados.
processo independente - quando são criados, vivem e morrem sem interagir entre eles , ou processos cooperativos - que partilha informação ou comunicam de forma explícita, pressupõe-se que partilham um endereço, ou por um canal de comunicação.
Dado isto os processos independentes competem entre si para ter acesso ao recurso , é responsabilidade do OS assegurar a atribuição de maneira

controlada para que não haja perdas. Requer, então, que apenas um único processo tenha acesso ao recurso de cada vez , causando assim exclusão mútua.



O caso dos processos cooperativos que partilham informação ou comunicam entre si, é responsabilidade dos processos envolvidos assegurar o acesso a zona partilhada seja feita de maneira controlada. Requer que apenas um único processo deva ter acesso ao recurso de cada vez, exclusão mútua. Enquanto ao acesso ao canal de comunicação deve ser visto como uma competição ao recurso comum.