Lista para Revisão - P1 DSID

1. Se você tivesse que escolher apenas uma característica fundamental de um sistema distribuído, qual escolheria ? Por que ?

Waldir:
Transparência. Segundo Tanembaum, um sistema distribuído é um conjunto de computadores independentes que trabalham em conjunto funcionando como um sistema único e coerente. Dado isso, caso a transparência não existisse, a própria ideia de sistemas distribuídos não existiria, pois a transparência garante que o SD irá se mostrar ao cliente como um único sistema, sendo que ele pode nem mesmo saber que está utilizando um SD.

Marcelo:
A característica fundamental de um sistema de informação distribuído é a transparência da distribuição, pois é a propriedade de gerar a ilusão de um ser um único sistema coerente.

2. Qual a relação entre comunicação e coordenação e porque ambas são fundamentais para um sistema distribuído ?

Waldir:
Há uma intrínseca relação entre comunicação e coordenação. Comunicação diz respeito a como os diferentes nodos do sistema distribuído se comunicam entre si e a coordenação garante sincronia nesta comunicação, fazendo com que as diferentes partes rodem na ordem correta, promovendo o funcionamento correto do SD evitando inconsistências.

3. O que a existência de relógio global implica para a execução de um sistema de software?

Waldir:
Em um sistema de software(centralizado), ter um relógio global significa garantir a sincronia entre os diferentes processos. Ex: para obter a hora, os processos a e b fazem uma chamada ao sistema operacional, se A executa antes de B ,então tA < tB para que ele seja executado antes.
Abel: (complementando…)
O que significa um possível impedimento para os sistemas distribuídos, uma vez que a centralização, geralmente, implica na geração de gargalos.

4. O que a ausência de relógio global implica para a execução de um sistema distribuído ?

Waldir:
A ausência de um relógio global em um sistema distribuído implica que cada computador que compõe o SD possui o seu próprio relógio, podendo causar um atraso ou adiantamento em relação aos demais. Por este motivo, os relógios devem ser sincronizados, garantindo a correta coordenação e comunicação descrita na questão anterior.

Julia:
A ausência de um relógio global em um sistema distribuído implica que cada computador que compõe o SD possui o seu próprio relógio, podendo causar um atraso ou adiantamento em relação aos demais. Por est e motivo, a **comunicação entre os nós** do sistema distribuído deve ser sincronizada, garantindo a correta coordenação e comunicação descrita na questão anterior.

Thiago:
Implica na necessidade de se utilizar mecanismos de sincronização para garantir a ordem correta de funcionamento de cada componente do SD.

Fabio:
A ausência do relógio global em SD's faz com que uma série de inconsistências sejam evitadas, uma vez que seria praticamente impossível e custoso sincronizar o relógio de cada máquina com um relógio global geral. Isso causaria problemas, pois mesmo que os relógios fossem em sincronia, os relógios individuais de cada componente podem ser executados em uma taxa ou granularidade diferente, fazendo com que eles fiquem fora de sincronia somente após um ciclo de clock local.

5. O que o desenvolvedor de um sistema distribuído deve fazer para contornar as limitações apresentadas no cenário da questão anterior ?

Waldir:
Deve procurar soluções para a sincronização dos relógios dos diferentes nós. Há diferentes algoritmos para sincronizar os diferentes computadores que compõe o sistema distribuído, alguns deles são:
Algoritmos centralizados (com servidor):
- Servidor passivo (Cristian)
- Servidor ativo (Broadcast e Berkeley)

Algoritmos distribuÌdos (sem servidor)

- Média Global
- Média local
.

6. Quais são as relações de compromisso (trade-offs) envolvidas na construção de uma visão coerente única de um sistema distribuído?

Kenji: em uma construção de um SD deve ser levado em conta certos compromissos, que ditam a respeito do custo que essa implementação irá gerar, a homogeneidade do sistema, isto é, o quão homogêneo serão os componentes que farão parte desse SD e a própria distribuição de recursos.

Paulo:
Trata-se do compromisso que se assume com a transparência a falhas, onde, apesar de aparentar ser um sistema único ele deve ser expressamente coerente ao usuário (seja em casos de sucesso ou de falha).

Juliana: as trade-offs, ou seja compensações (negociação envolvendo os objetivos) se relacionam aos ganhos e perdas a partir das decisões sobre as características do sistema distribuído. Por exemplo, um excesso de transparência pode comprometer o desempenho. Este deve ser considerado durante a busca por uma visão única do sistema distribuído e decidir quanto devemos priorizá-lo implicará na negociação dos objetivos (trade-offs).

7. Considerando que o middleware ele próprio é um sistema distribuído, qual é a diferenciação com o sistema distribuído do usuário? E com o sistema operacional?

Jordana:
Assumindo que por usuário entende-se o desenvolvedor de aplicações distribuídas:
O middleware caracteriza-se como um software provedor de uma interface entre a camada da aplicação e a camada de Sistema Operacional. Em outras palavras, o middleware é para um sistema distribuído o que o Sistema Operacional é para o hardware, porém sua atuação é na rede. O middleware é um gerenciador de recursos e apresenta um conjunto de funções e componentes que facilitam essa comunicação (entre a aplicação e as máquinas) e que justamente por essas características elimina a necessidade de implementações específicas para a aplicação (dependentes de cada hardware e SO onde ela será executada). O sistema distribuído do usuário é um sistema desenvolvido para atender determinados objetivos de seus idealizadores (seja uma empresa, uma pessoa ou qualquer outra entidade) e que, dependendo de sua robustez e necessidades, pode

depender de um middleware. A aplicação do usuário é executada acima do middleware e se utiliza das funcionalidades que este apresenta.

Kenji: complementando, um middleware não possui privilégios e nem controle sobre diversas funções, como processos bloqueadores gerenciados pelo SO, assim como o gerenciamento de memória dos processos criados que são gerenciados também pelo SO, que realiza um swap quando necessário para trocar espaços entre memória principal e secundária. Um outro exemplo é o gerenciamento pelo SO de concorrência entre os processos, coisa que a nível de middleware não é possível, já que o mesmo no máximo gerencia threads.

8. "Não é porque é possível construir sistemas distribuídos que isso necessariamente seja uma boa ideia" (tradução livre). Discuta essa frase relacionando os objetivos de oferecer um sistema distribuído.

Marcelo:
Um sistema distribuído é composto por máquinas que estão distribuídas ao longo da rede. Existe um custo a se pagar para fazer esse conjunto de máquinas aparentar ser um único sistema coerente, isto é, prover transparência da distribuição (uma das metas). A abertura de um SD também é uma meta e possui um custo de possivelmente sacrificar desempenho, uma vez que implica que serviços devem seguir uma interface comum. Em resumo existe um custo associado ao sistema distribuído e antes de usar um SD é necessário pensar se esses custos são maiores ou menores do que o benefício. Nos casos em que se necessita de um alto desempenho um SD pode não ser uma boa ideia, por exemplo.

Paulo:
Considerando a meta de "acesso a recursos" é possível que tenhamos problemas com rastreabilidade de comunicações e a eventual comunicação indesejável, uma vez que os recursos são compartilhados entre os nós da rede. No que diz respeito à "transparência", atingir completa transparência é muito difícil e não necessariamente trará apenas benefícios ao sistema: o usuário que tem ciência de que existem vários pontos potenciais de falha no sistema que podem ter impacto é mais complacente com o sistema que usa e passa a adotar melhores estratégias de uso (em melhores horários, por exemplo).
Na meta de "abertura" é muito complexo garantir que todos os componentes mantém as mesmas políticas de acesso e, eventualmente, isso pode prejudicar o desempenho do sistema. E, por fim, quanto à "escalabilidade" é um problema a possível inconsistência de atualizações dentro do sistema distribuído já que não há um relógio global e a confiabilidade da comunicação

Portanto, em casos em que se necessita alto desempenho, confiabilidade altíssima ou segurança muito rígida pode não ser conveniente desenvolver um sistema distribuído.

9. Em relação aos objetivos relacionados na questão anterior, discuta sobre o custo associado em alcançar cada um deles.

Marcelo:
Com relação as transparências os custos variam de acordo com cada tipo. Para se gerar a transparência de replicação e de localização é necessário usar nomes lógicos para os processos e recursos, o que implica a necessidade de criar um mecanismo de associar um nome lógico ao endereço físico e de manter esse mecanismo consistente. A transparência de acesso pode implicar em perda de desempenho pois esconder as diferentes formas de acesso pode exigir conversão da representação dos dados (uma máquina pode ser little endian e outra big endian).
No caso da abertura do sistema é fundamental seguir uma interface comum, o que nem sempre é bom já que pode implicar em sacrifício de desempenho.
A escalabilidade também é custosa. Imagine um sistema que executa em uma rede local. Nesse caso é mais vantajoso para esse sistema usar uma comunicação síncrona, entretanto, se o sistema foi desenvolvido pensando na escalabilidade outro tipo de comunicação menos vantajosa para uma rede local pode acabar sendo escolhido.

10. Diferencie portabilidade de interoperabilidade.

Ricardo:
Portabilidade se refere a capacidade de uma aplicação feita para um sistema A, poder rodar em um outro sistema B, sem modificações. Enquanto interoperabilidade se refere a capacidade de duas implementações de sistemas diferentes conseguirem coexistir e trabalhar em conjunto, usando um padrão comum de comunicação.

Kenji: complementando, a primeira dita princípios adotados em um SD pervasivo, na qual o sistema se adequa a certas variáveis como executar em diferentes dispositivos móveis e suas configurações, enquanto a segunda trata, em outras palavras, a respeito da transparência para que independente de quem vai executar as operações, as mesmas sejam transparentes e retornem o resultado esperado pela requisição.

11. Qual a relação entre a escalabilidade geográfica e o tipo de comunicação adotada em um sistema distribuído?

Marcelo:
A escalabilidade geográfica tenta fazer com que as distâncias entre os recursos não sejam facilmente percebidas pelo usuário. O tipo de comunicação usado influencia fortemente no cumprimento dessa meta. Por exemplo, quando os recursos estão muito longes uns dos outros, uma comunicação síncrona implicará num tempo de espera maior por parte do usuário. Nesse caso a comunicação assíncrona conseguiria esconder melhor essa distância pois enquanto se espera a resposta requisitada o computador do cliente pode adiantar alguma coisa, o que ameniza esse delay.

12. Os mecanismos de comunicação estudados proveem transparência ? De que tipos e em que níveis ?

Marcelo:
Sim.
No caso de RPCs e objetos remotos as transparências de localização e de acesso são fornecidas em um grau alto, uma vez que o cliente não se dá conta que o código que será executado está em outra máquina e que o stub faz a devida conversão nos tipos de dados para que a outra máquina possa interpretar corretamente os dados (novamente o exemplo da representação binária little endian e big endian).
Da mesma forma para MOM temos essas transparências, os roteadores cuidam da transparência de localização e os brokers da transparência de acesso.

Juliana: (Complementando) No caso do MOM, um cliente pode enviar e receber mensagens de qualquer outro cliente de maneira assíncrona, por estarem à conectados a um agente especial que fornece facilidades para criar, enviar, receber e ler mensagens (middleware). Por isso, o emissor e receptor não precisam estar sincronizados, nem precisam ser previamente conhecidos e a transparência é garantida por não existir uma ligação direta entre as aplicações.

13. Recentemente foi gerada a primeira imagem de um horizonte de eventos através da coleta de dados obtidos por diversos telescópios distribuídos no globo. Instrumentos científicos interligados que coletam dados em experimentos na ordem de petabytes para cientistas de diversas instituições de pesquisa do globo são organizados em sistemas distribuídos conhecidos como grades computacionais. Discuta os desafios enfrentados para alcançar escalabilidade nesse contexto.

Marcelo:
Nesse caso existem vários domínios administrativos, o que dificulta a escalabilidade administrativa. Cada domínio pode ter suas próprias políticas de segurança e uso dos recursos, o que leva ao problema de como conciliar essas políticas.
Da mesma forma a escalabilidade geográfica é um grande desafio pois a comunicação é ainda mais importante já que uma grande quantidade de dados está a disposição.

14. Quais são as técnicas para alcançar os tipos de escalabilidade discutidos na questão anterior ?

Jordana:
No caso da escalabilidade geográfica, pode-se pensar em 2 alternativas:
A primeira seria objetivar esconder a latência de comunicação na rede. Tal tarefa pode ser atingida através do uso de comunicação assíncrona, na qual um nó que realizou uma requisição não fica bloqueado até que chegue uma resposta (de um outro nó que pode estar do outro lado do planeta e, portanto, ainda mais sujeito a atrasos e perdas na rede) do nó solicitado.
A segunda seria utilizar replicação dos componentes do sistema. A replicação traz 2 vantagens principais: aumenta a disponibilidade do componente e possibilita balanceamento de carga, o que geralmente implica ganho de desempenho da aplicação.
No caso da escalabilidade administrativa pode-se pensar em particionamento e distribuição de componentes. Essa técnica é utilizada no DNS, sistema em que existe uma divisão por domínios (uma estrutura de árvore) que são divididos por zonas não sobrepostas.

15. Caracterize os seguintes sistemas distribuídos:
a) cluster computing
Waldir:
É um conjunto de computadores homogêneos (no geral possuem a mesma arquitetura, organização e sistema operacional) conectados via lan (local area network).

São utilizados para sistemas que requerem alta performance e disponibilidade, ou seja, se um nodo falha, todo o sistema continua funcionando a partir do balanceamento de carga entre os diferentes nodos restantes.

b) grid computing

Gabriel:

Grid computing é aquele que costuma ser montada como uma federação de computadores, na qual cada sistema pode cair sob um domínio administrativo diferente e pode ser bastante diferente no que se diz a hardware, software e tecnologia de rede usada

Bruno: (complementando)... O grid computing possui algumas características específicas, uma delas é que os recursos coordenados não são sujeitos a um controle central, havendo uma espécie de ambiente cooperativo (uso do tempo ocioso de processamento de um outro sistema, que por exemplo, está em um fuso horário diferente e não está em seu pico de uso). A utilização de padrões abertos de protocolos e interfaces é uma característica também muito comum nos sistemas de Grid Computing.

c) cloud computing

Kenji: cloud computing é a conhecida computação em nuvem. Serviços como esse são cada vez mais comuns, como a AWS da Amazon. Basicamente estamos falando de computadores, estejam eles em uma única rede ou não, que operam em um local geográfico diferente do acessado, possibilitando maior disponibilidade. Seus sistemas podem ser homogêneos ou não.

Abel: (complementando…)

Os recursos costumam ser disponibilizados sob demanda, para compra ou não, geralmente podendo ser separados nos seguintes tipos:

- IaaS: Infrastructure as a Service, cobrindo as camadas de hardware e infraestrutura;
- PaaS: Platform as a Service, cobrindo a camada de plataforma;
- SaaS: Software as a Service, onde aplicações são disponibilizadas.

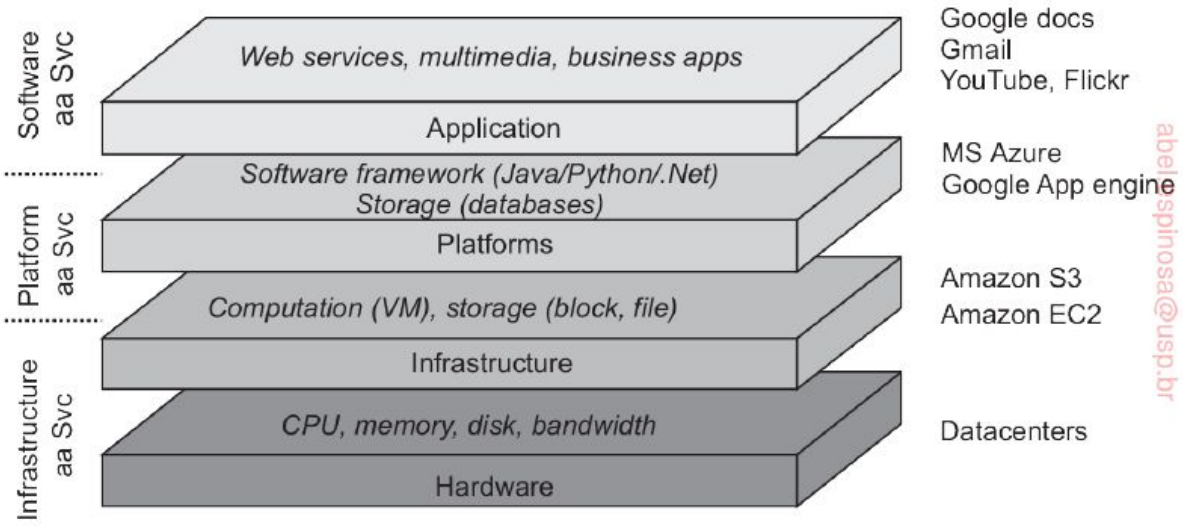

Figure 1.9: The organization of clouds (adapted from Zhang et al. [2010]).

16. Suponha que você trabalha em uma empresa que está considerando contratar serviços em uma nuvem computacional. Que fatores levaria em consideração na migração das aplicações para nuvem?

Jordana:
Se a empresa possui servidores próprios (o que implica que houve investimento prévio e há custo de manutenção) e deseja migrar, provavelmente existe uma justificativa como custo de manutenção, escalabilidade, segurança e/ou mesmo repassar a responsabilidade pela infraestrutura a uma empresa cujo modelo negócio é focado na prestação desse tipo de serviço. Após me certificar dos motivos dessa decisão por parte da empresa, faria alguns cálculos conforme proposto por Hajjat (2010) e verificaria o custo dessa migração (para esse cálculo utilizam-se informações tais como quantidade de transações em uma determinada quantidade de tempo e consumo de rede, dentre outros).

17. O que significam acoplamento referencial e acoplamento temporal ? Como isso se relaciona às principais tipos de sistemas distribuídos ?

	Temporalmente Acoplado	Temporalmente Desacoplado
Referencialmente Acoplado	Direto	Mailbox
Referencialmente Desacoplado	Evento	Espaço de dados compartilhados

R1)acoplamento referencial significa que existe uma referência explícita na comunicação. Acoplamento temporal significa que a comunicação acontece de forma síncrona.(alguem completa como se relaciona pf)

Kenji: O acoplamento referencial indica o quão necessário é as partes envolvidas se conhecerem referenciando uma às outras, isto é, dita o quão necessário é ter conhecimento sobre um ID (identificação), ou quaisquer outros atributos para ditar que um determinado processo está se comunicando com outra parte. Por exemplo, em um processo via sockets na qual um cliente necessita saber o IP do servidor e isso é referenciado diretamente em sua comunicação, é um exemplo de forte acoplamento referencial, dado que foi necessário dizer muito explicitamente a identificação para que uma determinada requisição fosse atendida, enquanto que, por exemplo, uma requisição em uma pesquisa da Google não necessita que o cliente saiba qual servidor contém essa informação, basta que ele conheça alguém que possa encaminhar o pedido para o servidor final tratar, semelhante ao que um roteador de borda faria.

O acoplamento temporal dita o quão necessário é uma comunicação se estabelecer de forma síncrona, isto é, que ambas as partes envolvidas estejam operantes para se comunicar. Um acoplamento temporal forte é aquele cujo cenário se faz necessário que, para a comunicação se estabelecer, as duas partes precisam estar online. Um middleware interferindo para que a comunicação seja persistente e não transiente torna esse acoplamento temporal mais baixo, uma vez que o middleware armazenando os dados da requisição para ser entregue ao destinatário (persistente) faz com que não seja estritamente necessário que ambas as pontas estejam no ar.

Ambas as terminologias citadas influenciam na arquitetura e organização de um sistema distribuído, na qual os diferentes tipos são mais aconselháveis de acordo com o nível de acoplamento referencial ou temporal, dado que, por exemplo, em

uma estrutura cliente servidor, em geral possui alto acoplamento referencial e alto acoplamento temporal, enquanto que uma estrutura descentralizada P2P, ela pode ou não ser acoplada referencialmente e pode ou não ser acoplada temporalmente, depende de vários fatores como a presença de um middleware, a localização geográfica das máquinas envolvidas, etc.

18. Faz sentido falar que um sistema distribuído pode ter uma organização centralizada? Caso faça, explique o que significa "distribuído" e "centralizado" em cada um desses conceitos.

R1) Existem arquiteturas de sistemas distribuídos centralizadas, onde uma máquina central (servidor) implementa a maioria dos componentes de software, enquanto que máquinas remotas (clientes) podem acessar o servidor e seus serviços através de mensagens. Em uma arquitetura de sistema centralizada, o próprio servidor pode ser um sistema distribuído, um conjunto de componentes agindo de forma única e coerente aos olhos dos seus clientes.

19. Na figura 2.16 são apresentadas algumas possíveis configurações de um sistema distribuído cliente-servidor organizado em duas camadas. Apresente exemplos de sistemas para cada uma das situações (a) a (e).

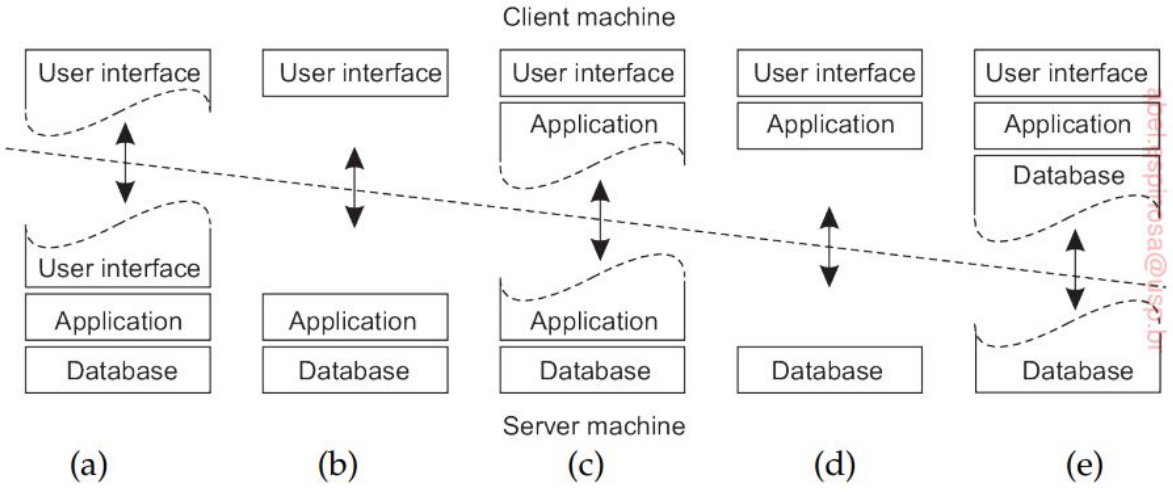

Figure 2.16: Client-server organizations in a two-tiered architecture.

a) Um sistema onde o lado do usuário seja apenas um terminal que envia comandos para um servidor que entende o pedido, acessa e processa os dados e retorna a resposta. O servidor também possui uma interface para gerenciamento de recursos.

b) Um sistema como a opção a), porém o lado servidor não possui interface alguma

c) A camada de aplicação é movida para o lado do usuário, permitindo que o sistema interaja com o usuário, por exemplo, para pedir que o mesmo corrija algum campo que tenha preenchido.
d) São casos onde o lado do cliente é um PC completo, por exemplo, e no servidor ocorrem todas as operações de ETL em dados.
e) Como o exemplo anterior, porém a aplicação no usuário mantém um lugar para guardar os dados, além do servidor.

Jordana:

a) **Terminal burro:** terminais burros são aqueles que não têm a capacidade de processar sequências de escape especiais que realizam funções tais como apagar uma linha, limpar a tela ou controlar a posição do cursor, ou seja, são muito limitados.
b) **Terminal X:** um Terminal X é um terminal de visualização e de entrada de dados para aplicações clientes do X Window System.
   Página web simples: aplicação que apresenta um front-end contendo apenas HTML e CSS, sem apresentar uma camada de lógica/regra de negócio e, que, portanto, tem a aplicação executada apenas no servidor. Exemplos: aqueles sites de prestadores de serviços que contém algumas páginas HTML simples e bonitas, mas que não permitem ao usuário interagir muito com a tela, sendo muito mais informativas.
c) **Aplicações web:** a maioria das aplicações web atuais são apresentadas nesse modelo. O Youtube, o tidia e o facebook são alguns exemplos.
d) **Internet banking:** o aplicativo desktop do Itaú, por exemplo.
   d.2) (Abel) Acho que um exemplo mais simples é de qualquer aplicação "tradicional" em que o banco de dados não esteja na mesma máquina. Um CRUD, por exemplo. Imagino que o Internet banking esteja muito mais pro item c).
e) **Web browser com cache:** Enquanto aplicação, o web browser concentra a interface de usuário, a aplicação e ainda armazena dados de navegação no próprio cliente.
   Aplicativo Mobile: App que utiliza um banco de dados como o sqlite (no celular) e funciona offline, realizando sincronização em alguns momentos, quando há conexão com a internet. No momento da sincronização ele poderia jogar os dados para um servidor, mas em um primeiro momento estavam todos no cliente.

   Obs. Nas aulas, a XL comentou essa figura e disse fica a dica para a prova… então cheguem manjando bem como dar exemplo de cada um deles...

20. Qual (quais) das organizações discutidas na questão anterior é(são) melhor(es) ? Porque?

As últimas, d) e e), por exemplo, são amplamente utilizadas no mundo corporativo. Mas não existe uma arquitetura melhor que a outra. É preciso escolher a que mais se enquadra no problema enfrentado.

Juliana: DEPENDE das necessidades (apresentar exemplos)

21. Discuta a relação dos sistemas peer-to-peer estruturados e não estruturados com o estabelecimento de redes overlay.

Kenji: Redes overlay são redes que possuem uma parte estruturada e outra parte não estruturada. Redes deste tipo são interessantes quando não temos que nos preocupar com a organização de uma parte dos dados ou como eles são tratados (busca, inserção, um CRUD, resumidamente) enquanto que, com certos dados obtidos da camada P2P não estruturada, se faz necessário organizar em uma outra topologia mais estruturada para que as respectivas informações e seu acesso sejam mais rápidos e precisos. Geralmente tal rede é implementada quando existe uma boa parte dos dados que não precisam ser frequentemente tratados e por isso não precisa se preocupar em organizá-los, o que diminui o seu custo e implementação, enquanto que existem outros dados mais usuais e eles precisam estar disponíveis de forma eficaz, porém tal eficiência em mantê-los organizados sobre alguma estrutura (anel Chord, CAN, etc) é custosa.

22. Como funciona o NFS ? Qual é a arquitetura desse sistema distribuído ?

Kenji: O NFS, da sigla Network File System, é uma topologia na qual clientes acessam arquivos segregados por diretórios no servidor, como um repositório compartilhado, cabendo ao servidor disponibilizar tais arquivos e gerenciar suas requisições. Dessa forma, o NFS se baseia em um estilo arquitetônico centrado em dados, e pode-se dizer que seu tipo de arquitetura é centralizada (cliente/servidor).

23. Como funciona a Web ? Qual é a arquitetura desse sistema distribuído ?

Kenji: Sistemas distribuídos baseados na Web são um padrão conhecido por muitos, na qual estamos falando basicamente de requisições cliente/servidor via browser, utilizando o protocolo HTTP. Sua arquitetura é centralizada, uma vez que quem pode tratar a requisição do cliente é somente o servidor.

24. Quais são as vantagens que uma implementação multi thread tem sobre uma mono-treaded em:
a) sistema não distribuído

Kenji: uma implementação multi thread em um sistema único, isto é, que opera em uma única máquina isolada, só faz sentido se pensarmos em paralelismo, dado que as requisições a serem tratadas não seriam encaminhadas para ninguém e seria diretamente encaminhadas para ele, que deve processar e retornar uma resposta o mais rápido possível, tendo grande vantagem sobre uma mono thread.

Marcelo: complementando, é possível continuar a execução mesmo após o início de uma operação bloqueante, como um acesso ao disco por exemplo.

b) sistema distribuído

Kenji: em um sistema distribuído, a utilização de multi threads também oferece paralelismo, porém deve ser bem estruturada, uma vez que os processos a serem tratados podem possuir finalidades e destinos diferentes, além do que, se implementarmos multi thread somente em uma parte do sistema, como no middleware, pode engargalar os demais componentes do sistema. Uma boa estrutura para tratar um load balance, como um sistema em cluster, além de logicamente prevenir indisponibilidade, pode auxiliar no processamento múltiplo.

Jordana: (complementando a resposta anterior, apenas): Em um servidor, por exemplo, se não há multithreads, a cada requisição que chegar ao servidor haverá o bloqueio total do servidor até que se conclua a comunicação/requisição. Já quando se tem multithreads, uma possibilidade é existir uma thread despachante que distribui requisições entre as demais existentes e garante que mais de uma tarefa seja realizada simultaneamente e possibilita que uma chamada bloqueante interrompa apenas uma thread e não o servidor inteiro.

25. As implementações dos sistemas que seguem as organizações descritas na Questão 19 podem ser multithreaded ? Explique se isso é possível e se é vantajoso nas situações de (a) a (e).

Jordana:
Achismo puro (vou tentar confirmar com a Gisele):
Em C: com certeza! Aplicações web comuns como Youtube e Facebook funcionam dessa forma. As páginas HTML são carregadas em etapas, sendo que as informações mais leves (HTML contendo textos simples, por exemplo) costumam ser carregadas e mostradas antes para o usuário (no cliente, nesse caso, navegador), enquanto imagens e vídeos e demais dados pesados são enviados posteriormente. Cada tipo de arquivo dentro da página HTML costuma ser de responsabilidade de uma thread diferente.

Em E: um aplicativo mobile pode apresentar funcionalidades que se beneficiem de multithreading. Um exemplo seria um aplicativo que utiliza mapas e realiza cálculos de áreas, rotas, dentre outros, por exemplo, pois nesse tipo de atividade os cálculos muitas vezes podem ser paralelizáveis.

Kenji: d) em sistemas fat client (clientes gordos), como exemplificado a plataforma do Internet Banking, não só pode ser multithread como é praticamente necessário, pois diversos processos devem acontecer simultaneamente, como o timeout da sessão do cliente logado ao mesmo tempo em que busca os contatos favoritos e ainda carrega as informações para iniciar uma transferência.

26. "A era dos mainframes dependia muito da virtualização. Hoje ela é fundamental para sistemas distribuídos no século 21." Discuta essa afirmação mostrando como e onde se dá o emprego da virtualização atualmente.

> *Atualmente, software de baixo nível se desenvolve mais rápido que software de baixo nível, criando um gargalo, onde as interfaces, que acabam virando legado, não são compatíveis com as novas plataformas na qual elas mesmo rodam. Esse gargalo é resolvido através da virtualização*

27. É possível combinar algum dos quatro paradigmas de mobilidade de código com as organizações descritas na Questão 19? Explique como nas situações de (a) a (e).

Kenji: As quatro mobilidades mencionadas se tratam das variações de mobilidade fraca/forte e mobilidade iniciada pelo remetente/pelo destinatário.

28. Quais são os desafios para migração de código em sistemas distribuídos heterogêneos?

Kenji: quando tratamos de migração de código sempre temos que levar em conta obviamente a compatibilidade do código de origem para o destino. Quando pensamos nisso, em sistemas distribuídos heterogêneos temos que levar em conta se o destinatário possui suporte para o código como ele está, o que implica por exemplo em coisas simples como separadores de arquivo em um path, pois separadores são distintos em Windows e Linux. Muitas vezes a migração não é tão transparente e é necessário realizar ajustes para tal, seja tornando tais configurações parametrizáveis em arquivos xml ou criando uma API intermediária com uma linguagem em alto nível via scripts ou outros exemplos que possam tratar os dados para que o devido processamento ocorra. Tais scripts podem ser realizados em Python por exemplo.

29. A arquitetura de uma solução geralmente temos as seguintes camadas básicas: hardware, sistema operacional, middleware e aplicação.
a) O que é considerado camada de aplicação para sistemas distribuídos ?

Kenji: vou pesquisar mais a respeito, mas de forma mais ampla, a camada de aplicação em SD não inclui o middleware, na qual não existem protocolos com tratamentos elaborados na camada de usuário. Meio difícil delimitar até onde um middleware pode ir, mas em geral, isso é definido no contexto do SD em questão, se o tratamento é específico do negócio tratado, então este deveria estar na camada de aplicação e não no middleware, pois o middleware deve ser algo que seja mais genérico, com tratamentos que não se fixam em determinados formatos de arquivo por exemplo ou se vem em determinado encoding. Em um sistema Web por exemplo, o uso do browser é o que eu trato como camada de aplicação, porém, tal browser também cuida de certos pontos como segurança (principalmente no IE), acho algo muito subjetivo, se alguém puder complementar, é nois.

b) O que é considerado camada de aplicação para redes de computadores ?

Jordana:
(Wikipedia) A camada de aplicação é um termo utilizado em redes de computadores para designar uma camada de abstração que engloba protocolos que realizam a comunicação fim-a-fim entre aplicações. Exemplos de aplicações são: HTTP, SMTP, FTP, SSH e Telnet.
Nota-se que um dos modelos arquiteturais utilizados, a RESTful se baseia no protocolo HTTP, utilizando-se diretamente das suas convenções (POST, GET, DELETE e PUT)

Kenji: só para complementar, acho que o mais interessante em expor nessa questão é que o middleware faz parte da camada de aplicação para a rede de computadores, enquanto em um sistema distribuído o middleware se difere, dado que nesse outro contexto, a camada de aplicação trata sobre a camada do usuário somente, algo muito mais superficial.

c) Onde começa e termina a camada de middleware ?
Julia:
O middleware é uma camada de software logicamente localizada entre o sistema operacional e as aplicações distribuídas. Dependendo do sistema distribuído, a camada de middleware pode ser mais gorda ou mais magra..

30. Foram estudados alguns mecanismos de comunicação que provém transparência de acesso.Descreva cada um e discuta qual deles é o melhor.

Kenji: de cabeça penso em RPC como um exemplo de transparência de acesso. Alguém sugere outros mecanismos?

31. Apresente exemplos de aplicações que tenham o requisito de comunicação:

a) transiente síncrona

Kenji: uma aplicação de transmissão de mídia sem buffer, como um streaming real time (live), na qual ambas as partes devem estar sincronizadas (propagador da transmissão e receptor da imagem/vídeo) e não é persistido a transmissão, de forma que caso o cliente caia ou pause a transmissão, ao voltar, o mesmo passa a receber a transmissão online e não continua de onde parou.

b) transiente assíncrona

Kenji: uma requisição web, como uma pesquisa no Google. O servidor pode estar fora ou não estar acessível (você está sem internet) mas é possível mandar a requisição (pesquisa) porém ela obviamente não será respondida com o devido resultado e, por mais que você volte a ter Internet ou que o servidor volte a responder, é necessário enviar outra requisição (um F5 no browser por exemplo), pois sua requisição não foi persistida (transiente).

c) persistente síncrona

Kenji: semelhante ao item a), uma transmissão de mídia em fluxo na qual as pontas participantes devem estar disponíveis, porém, com um serviço de buffer para armazenar os dados já transmitidos (pausar e continuar) ou recuperar algum dado já transmitido (voltar para algum ponto do vídeo já transmitido) faz desta um exemplo de aplicação persistente e síncrona.

d) persistente assíncrona

Kenji: um chat, aplicativo de conversação como o Whatsapp, na qual um middleware se encaminha de armazenar a mensagem que você enviou mesmo que o seu contato não esteja online para encaminhar ao mesmo assim que possível.

32. Em relação à questão anterior, como é feita a implementação de cada tipo de serviço de comunicação ?

33. Agora é sua vez de apresentar um problema em que se aplicariam conceitos e técnicas vistos até o momento na disciplina e também discutir sua proposta de solução.

a) Apresente de maneira geral o problema, relacione os objetivos do sistema em relação aos níveis de transparência, escalabilidade, desempenho, compartilhamento, abertura.

b) Proponha uma arquitetura, detalhando decisões de projeto que visam alcançar os objetivos e requisitos apresentados em a)

c) Discuta detalhes relativos à implementação de processos (multithreaded ou não, com migração de código ou não, etc.) e também de comunicação. Não se esquecendo de que a solução vai além de implementar o nível de aplicação, mas também integrá-lo aos demais níveis (middleware, SO e hardware).

PROVA ANO PASSADO

Questão 1:
Perguntava sobre sistemas escaláveis:
a-) definição de sistemas escaláveis
Paulo:
Ser escalável é uma característica de um sistema distribuído onde:
-> Vários usuários podem ser adicionados sem nenhuma queda considerável no desempenho do mesmo;
-> Os usuários podem estar muito distantes uns dos outros sem que isso impacte na forma como se comunicam e nem crie delay nas comunicações;
-> A complexidade de sua administração (segurança, gerenciamento, etc.) não se modifica conforme cresce sua demanda.

Marcelo: tem uma pergunta parecida no livro e a resposta é a seguinte:
“A system is scalable with respect to either its number of components, geographical size, or number and size of administrative domains, if it can grow in one or more of these dimensions without an unacceptable loss of performance.”

b-)distribuições de um sistema escalável considerando um sistema distribuído pelo mundo todo

Paulo:

Um sistema distribuído pelo mundo todo deve garantir algumas transparências para seus agentes externos, de forma que não deva ser possível determinar a localização física de um recurso qualquer, não ocorra impactos no uso caso ocorra uma migração e que o usuário consiga, até mesmo, acessar recursos enquanto se desloca (de um ponto a outro) usando um laptop ou celular, por exemplo.

Kenji: complementando, deve também procurar transparecer a latência em suas requisições.

c-) Quais seriam as implicações de usar uma determinada distribuição para um sistema escalável

Paulo (falei sobre os tipos de escalabilidade, não ficou clara pra mim):

- Escondendo latências de comunicação: evitar ficar esperando respostas de serviços remotos sempre que possível (usando comunicação assíncrona);
- Particionando: dividindo determinado(s) componente(s) em pedaços menores e espalhando-os pelo sistema
- Replicação: aumentar a disponibilidade do re
  curso, seja criando cópias do componente ou usando *caching*, consequentemente pode gerar inconsistência de atualização.

Kenji: a questão quer saber quais as consequências ao adotar um determinado tipo de distribuição, assim como você descreveu na replicação, que ao adotar o mesmo tem a implicação de se preocupar em manter os registros replicados atualizados e consistentes.

Questão 2:

Perguntava sobre Middleware orientado a mensagens:

a-) O que é comunicação assíncrona e persistente

Marcelo: em uma comunicação assíncrona o cliente continua sua execução após a requisição ser submetida. Em uma comunicação persistente a mensagem fica armazenada em um mecanismo de armazenamento não volátil, o que implica que o remetente ou o destinatário não precisam estar em execução para a comunicação acontecer.

Em uma comunicação assíncrona persistente o cliente não fica bloqueado (ou seja, continua a execução após enviar a mensagem) e a mensagem não se perde quando uma das partes está inativa, o que permite que o cliente e o servidor sejam fracamente acoplados com relação ao tempo.

b-) Funcionamento do MOM (Middleware Orientado a Mensagens)

Marcelo:

A ideia é que as aplicações conversem através de mensagens que são retiradas e inseridas em filas locais. Cada aplicação lê mensagens de uma fila e coloca mensagens em outra fila, que são gerenciadas por um gerenciador de fila. Um gerenciador de fila é implementado através de um outro processo encarregado de gerenciar a fila ou através de uma biblioteca. Geralmente o gerenciador de fila e a aplicação ficam na mesma máquina, porém no pior caso fica na mesma rede local. Uma vez que a aplicação insere a mensagem na fila de envio a mensagem pode ser enviada para a fila de entrada do remetente.

É interessante que as filas possuam nomes lógicos para aprimorar a transparência de localização. Uma abordagem para isso é manter em todos os gerenciadores de fila uma lista relacionando cada nome lógico com o endereço físico, o que pode levar ao problema de manter a consistência em todos os gerenciadores fila. Uma maneira de solucionar esse problema é a utilização de gerenciadores de filas especiais chamados de roteadores cuja função é repassar a mensagem para o destinatário. Dessa forma manter a consistência das listas de endereços fica mais fácil já que não será necessário colocar essas listas em todos os gerenciadores de fila, apenas nos roteadores.

Os brokers são gerenciadores de fila cuja função é traduzir a mensagem do remetente para um formato que o receptor entenda. Geralmente os brokers não são considerados uma parte integral do sistema, mas apenas uma outra aplicação.

Então quando a aplicação A deseja enviar um mensagem para a aplicação B é necessário que a aplicação A coloque a mensagem em sua fila local e que o seu gerenciador de fila faça o envio. A mensagem pode acabar passando para outro gerenciador de fila, no caso um roteador, que traduzirá o nome lógico em um endereço físico. Antes de chegar na aplicação B a mensagem pode passar por um broker, que será responsável por traduzir a mensagem de A de um maneira que B possa entender. B poderá ler a mensagem quando ela estiver em sua fila de entrada. Neste ponto é interessante notar que A não precisa estar em execução.

O gerenciadores de fila fazem parte do middleware desse sistema.

c-)

Questão 3:
Perguntava sobre arquitetura Restful
Paulo:
A arquitetura Restful vê o sistema distribuído como uma coleção de recursos gerenciados por componentes, tais recursos podem ser manipulados por aplicações remotas. São características:

- Recursos são identificados usando um único esquema de nomeação;
- Todos os serviços oferecem a mesma interface;
- Mensagens enviadas de/para um serviço são auto-descritivas;
- Após a execução de uma operação em um serviço, o componente esquece tudo sobre quem chamou a operação.

Questão 4:
Exemplo de aplicações que tinham uma camada (de interface de usuário, de aplicação ou de dados) quebrada entre cliente e servidor ou então presente apenas no cliente ou no servidor

Questão 5:
Vantagens e desvantagens de usar RPC ao invés de APIs de Sockets
Marcelo:
RPC fornece um maior grau de transparência de acesso, enquanto no casos de sockets é necessário implementar a tradução da mensagem. A vantagens e desvantagens são relativas ao problema (o famoso depende). Em uma situação em que não se deseja deixar a cargo da aplicação a conversão das mensagens o RPC acaba sendo uma alternativa melhor. Quando se deseja que a aplicação tenha mais liberdade para escolher o protocolo de comunicação a ser usado o RPC entra em desvantagem. Em resumo, tudo depende da flexibilidade que se deseja dar para a aplicação. RPC provê mais facilidade de uso e menos flexibilidade, enquanto sockets fornecem menos facilidade e maior flexibilidade.

Caparelli:
Complementando a resposta anterior: o uso da API de Sockets permite a construção de um sistema utilizando um protocolo da camada de aplicação bem definido e difundido (como HTTP), permitindo maior interoperabilidade, compossibilidade e extensibilidade de um sistema ao adotar uma abordagem que favorece o uso de interfaces abertas.

Kenji: complementando, basicamente estamos comparando um protocolo de baixo nível (socket) com um de alto nível (RPC). Assim como foi falado, se trata de escolher qual se melhor adapta para as condições da sua aplicação. Existem casos de RPC utilizando sockets ainda, já que o uso dos sockets se limita praticamente em estabelecer conexão com o servidor.