Estratégia Concursos

Aula Única

BNB (Especialista Técnico - Analista de Sistemas Perfil 1: Desenvolvimento de Sistemas) Servidores de Aplicação

Autor:
Evandro Dalla Vecchia Pereira

20 de Março de 2024

Índice

Sistemas Distribuídos

Vamos começar o assunto com a definição de sistemas distribuídos, segundo dois autores conhecidos. Silberschatz define um sistema distribuído como:

"Uma coleção de processadores que não compartilham a memória ou um clock. Em vez disso, cada processador possui sua própria memória local. Os processadores se comunicam entre si através de diversas redes de comunicações, como barramentos de alta velocidade".

Tanenbaum define que: "Um sistema distribuído é um conjunto de computadores independentes entre si que se apresenta a seus usuários como um sistema único e coerente."

Resumindo, são sistemas que o usuário "enxerga" como um único computador, mas na verdade existem alguns (ou muitos), geograficamente distribuídos. Como as mensagens são trocadas e como é realizado o processamento, o usuário nem fica sabendo (é transparente), ele só recebe o resultado final.

Imagine uma situação em que são necessários diversos cálculos para resolver um problema, como por exemplo, verificar se há sinais enviados através de estrelas ou planetas (vida extraterrestre). Então, quem quiser participar desse projeto, pode ceder processamento de seu computador quando estiver ocioso (através de um programa previamente instalado). Os computadores voluntários realizam os cálculos e enviam resultados a outros computadores. Parece loucura, não? Mas existe esse projeto, é só verificar em <https://www.seti.org/>.

Note que o exemplo acima, que é real, mostra que para um usuário se trata de um único sistema, mas "por baixo dos panos" diversos computadores interagem, realizam cálculos etc. Pode ser que um computador possua processador RISC, outro CISC, um com sistema operacional Unix, Linux, Windows, Mac OS, e por aí vai. No fim, todos se entendem (de forma transparente) e o usuário vê apenas uma interface.

Quando falamos em transparência, não é apenas em relação à localização. Vamos ver na tabela a seguir que um sistema distribuído pode requerer alguns tipos de transparência.

Transparência	Descrição
Acesso	Oculta diferenças na representação de dados e no modo de acesso a um recurso.
Localização	Oculta o lugar em que um recurso está localizado.
Migração	Oculta que um recurso pode ser movido para outra localização.
Relocação	Oculta que um recurso pode ser movido para outra localização enquanto em uso.
Replicação	Oculta que um recurso é replicado.
Concorrência	Oculta que um recurso pode ser compartilhado por diversos usuários concorrentes.
Falha	Oculta a falha e a recuperação de um recurso.

Os sistemas distribuídos são muito comuns em ambientes que exigem alta disponibilidade, pois mesmo, que parte dele "caia", o sistema ainda continua operando. Em comparação a um sistema centralizado, as principais vantagens são a possibilidade de crescimento incremental e a possibilidade de implementação de tolerância a falhas através da replicação de processos em unidades de computação distintas.

Vejamos alguns exemplos de sistemas distribuídos bastante conhecidos:

Buscadores:

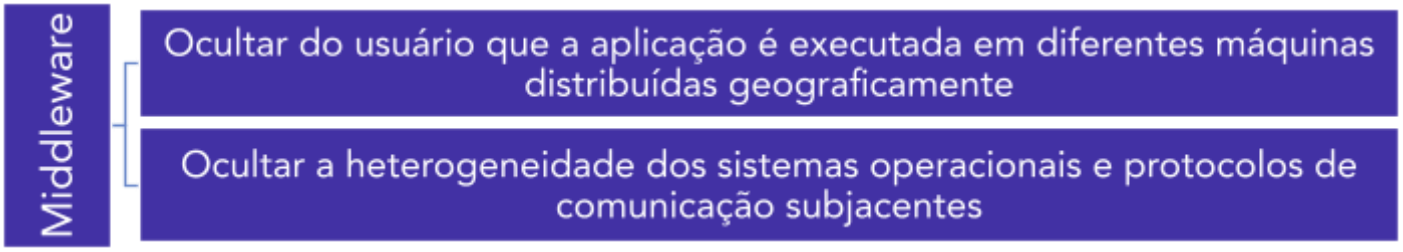
Internet:

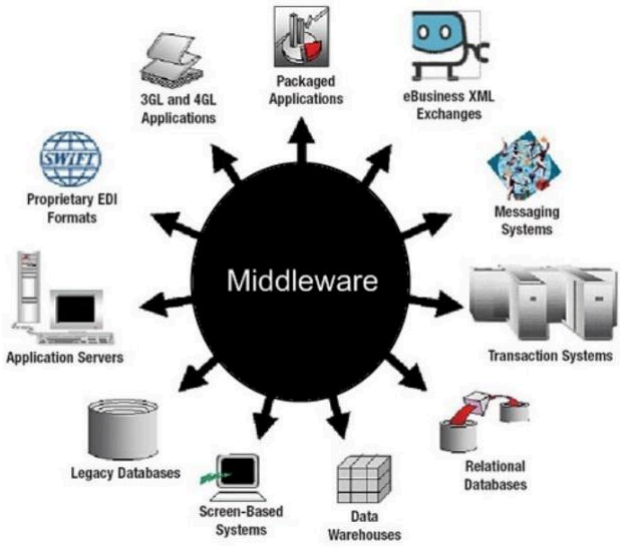
Computação em Nuvem:

Alguns conceitos recorrentes em provas de concurso são:

Middleware: camada de software localizada logicamente entre uma camada de nível mais alto (usuários e aplicações) e uma camada subjacente (frameworks e facilidades de comunicação). As principais funções da camada de middleware são:

- Ocultar do usuário que a aplicação é executada em diferentes máquinas distribuídas geograficamente;
- Ocultar a heterogeneidade dos sistemas operacionais (ex.: Windows, Linux etc.) e protocolos de comunicação subjacentes.



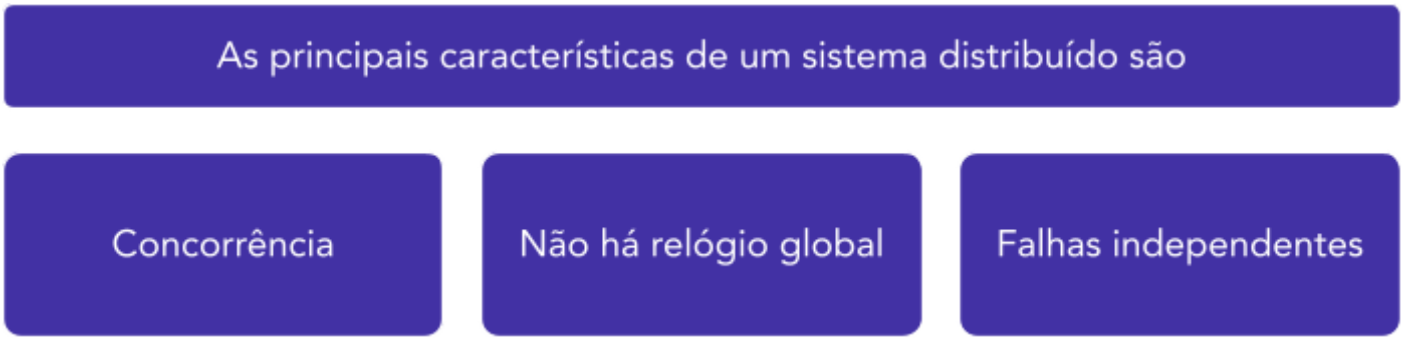


Processo: conjunto dos recursos alocados a uma tarefa para sua execução. Um processo é um programa em execução ou uma forma de gerenciar recursos. Cada tarefa precisa de um conjunto de recursos para executar e atingir o seu objetivo: processador, memória, dados, arquivos, conexões de rede etc.

Escalonamento: em muitas situações pode haver mais de um processo competindo pelo uso de um mesmo recurso. O escalonador do sistema distribuído é o responsável em decidir o nó de processamento responsável pelo processamento da tarefa. Portanto, trata-se de um dos componentes mais importantes, o qual utiliza um algoritmo de escalonamento.

As principais características de um sistema distribuído são:

- Concorrência: diferentes nós do sistema distribuído executam-se em concorrência;
- Não há relógio global: os relógios locais não estão necessariamente iguais, assim a única forma de coordenação é por troca de mensagens;
- Falhas independentes: qualquer componente pode falhar de forma independente de outros



As vantagens de sistemas distribuídos incluem a possibilidade de seu crescimento incremental (computadores e linhas de comunicação podem ser acrescidos ao sistema), a possibilidade de implementação de aplicações inerentemente distribuídas e a possibilidade de implementação de tolerância a falhas e a replicação de processos em unidades de computação distintas.

As principais motivações para a utilização de um sistema distribuído podem ser:

- Distribuição geográfica (organização com instalações em várias localidades diferentes);
- Necessidade de ligação entre organizações independentes;
- Necessidade de extensibilidade, modularidade ou crescimento gradual;
- Partilha de recursos como troca de informação entre departamentos e/ou empresas;
- Necessidade de maior disponibilidade ou replicação de dados;
- Necessidade de maior desempenho ou de distribuição de carga.

Parece que tem só coisa boa, heim? Não...vamos a alguns desafios:

- Segurança: intrusões podem acarretar na leitura de mensagens em trânsito e/ou a injeção de novas mensagens;

- Escalabilidade: o sistema deve continuar a funcionar de forma eficaz mesmo que haja um crescimento significativo no número de recursos e no número de clientes;

- Heterogeneidade: Equipamentos com representações de dados diferentes (arquiteturas RISC, CISC), sistemas operacionais diferentes (Windows, Linux etc.) devem ser integrados no mesmo sistema distribuído.

Funções, Modelos/Arquiteturas e Plataformas

Gerenciamento de recursos: a principal tarefa do sistema é garantir o controle sobre quem usa o quê. Esse controle também permite o compartilhamento dos recursos.

Gerenciamento de processos: essa funcionalidade distribui o processamento de forma justa entre as aplicações, evitando que uma aplicação monopolize o recurso. A gerência do processador está diretamente relacionada ao escalonamento. A sensação de que um processador está sendo compartilhado entre os vários processos é dita pseudoparalelismo.

Nos sistemas distribuídos podemos observar o paralelismo real, típico dos sistemas multiprocessados. O ciclo de processamento típico pode ser resumido da seguinte maneira:

- Requisitar: se a requisição não pode ser atendida imediatamente, então o processo requisitante deve esperar até obter o recurso;
- Usar: O processo pode operar sobre o recurso;
- Liberar: O processo libera o recurso.

Modelo cliente-servidor: os servidores mantêm recursos e servem pedidos de operações sobre esses recursos. Servidores podem ser clientes de outros servidores. Os próprios nomes ajudam a entender (servidor serve recursos e o cliente é quem solicita). Trata-se de uma arquitetura simples e permite distribuir sistemas centralizados muito diretamente, porém é pouco escalável, limitado pela capacidade do servidor e pela rede que o liga aos clientes. Ou seja, um único servidor não consegue atender milhares de clientes de forma satisfatória, por exemplo.

Modelo peer-to-peer: todos os processos possuem papeis semelhantes, sem distinção entre clientes e servidores. As principais características são mais ampla distribuição de carga (computação e rede), maior escalabilidade, o sistema expande-se acrescentando mais pares, e a coordenação mais complicada que cliente-servidor.

Componentes interagem entre si como "pares" por meio de chamadas/retorno numa rede. Não há hierarquia, como ocorre em um modelo cliente-servidor, e cada par pode interagir com qualquer outro. Exemplos de sistemas distribuídos em Peer-to-Peer (P2P) são abundantes, mas os atualmente mais conhecidos podem ser citados: BitTorrent, uTorrent, eMule, entre outros.

Vamos ver algumas plataformas que dão suporte aos sistemas e aplicações distribuídas:

- Java Messages: gestão de filas de mensagens da plataforma J2EE;
- MSMQ: sistemas de filas de mensagens da Microsoft;
- MQseries: sistemas de filas de mensagens da IBM;
- Active Enterprise: plataforma de integração da fabricante Tibco;
- Biztalk: Enterprise integration broker da Microsoft;

São inúmeras as soluções tecnológicas para implementar a distribuição:

- Utilização das interfaces de comunicação distribuída com *sockets*;
- Plataformas Cliente-Servidor;
- *Brokers* de Mensagens ou *Message Oriented Middleware*;
- Sistemas de Invocação Remota de Objetos (ex.: RMI ou CORBA);
- Sistemas Operacionais Distribuídos que permitam distribuição de todos os serviços do sistema;
- Web Services.

Cluster x Grid

Agora vamos ver um tópico BASTANTE ABORDADO em provas de concurso, os tipos de sistemas distribuídos: computação em cluster, computação em grade (*grid*) e computação em nuvem. Vamos focar nos dois primeiros, a seguir.

Clusters são sistemas de processamento compostos por uma coleção de computadores autônomos, interconectados e que trabalham em conjunto para processar uma tarefa.

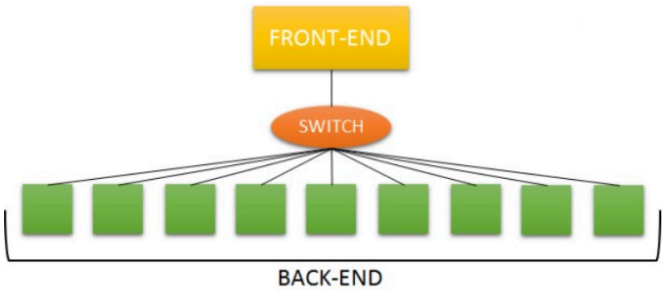
Tipicamente na computação de cluster é utilizada programação paralela na qual um único programa é executado em paralelo. Dependendo do objetivo que se pretenda, podemos ter os seguintes tipos de cluster:

- Cluster de alto desempenho: direcionado a aplicações bastante exigentes no que diz respeito ao processamento;
- Cluster de alta disponibilidade: tem como objetivo permanecer ativo por um longo período e em plena condição de uso. Consegue detectar erros e se proteger de possíveis falhas;
- Cluster de balanceamento de carga: controla a distribuição equilibrada do processamento. As tarefas de processamento são distribuídas o mais uniformemente possível entre os nós.

As principais vantagens da utilização de clusters são:

- custo-benefício: pode-se obter resultados tão bons quanto ou até superiores que um servidor sofisticado a partir de um conjunto de máquinas simples;
- escalabilidade: é possível aumentar a capacidade de um cluster com a adição de nós ou remover máquinas para reparos sem interromper a aplicação;
- facilidade de personalização para o atendimento da aplicação;
- existência de opções de softwares sem custos de licenciamento para cluster.

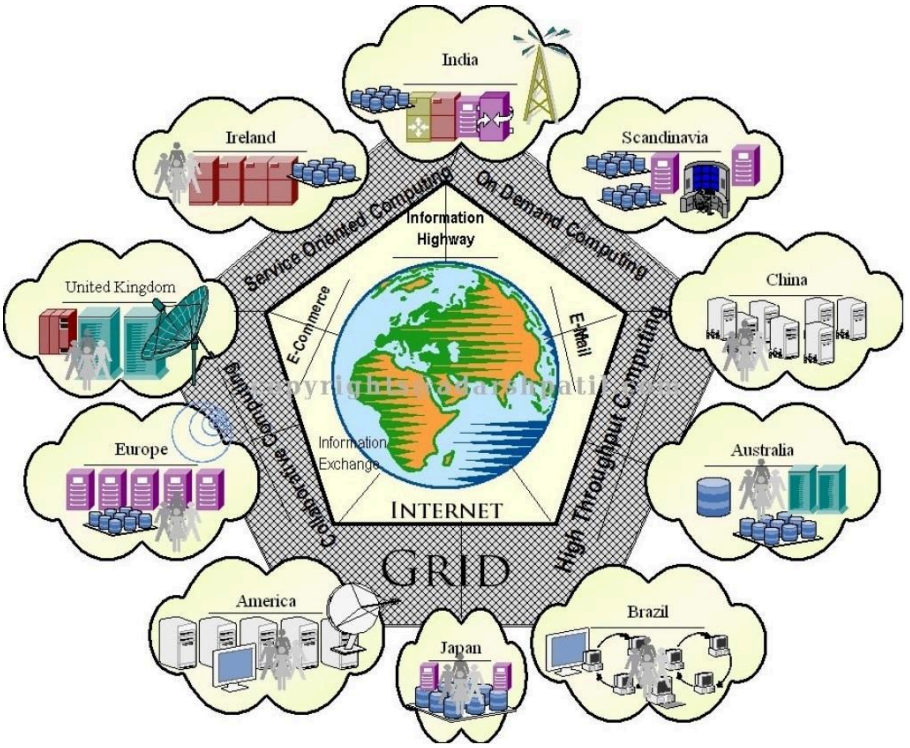
Resumindo, cluster é um agregado de computadores, um recurso de processamento utilizado em situações em que se exige a alocação exclusiva de um conjunto de recursos por longo período de tempo. Geralmente, existe um nó mestre que gerencia e divide as tarefas entre os demais nós (escravos).



Por exemplo, um software para tentar quebrar uma senha por força bruta pode ser paralelizado em um cluster, sendo que o mestre envia para cada escravo uma fatia das possibilidades de senha (um recebe a fatia de AAAA... até AZZZ..., outro recebe de BAAA... até BZZZ..., e assim por diante, se algum escravo descobrir a senha, avisa ao mestre para que tome as providências de cancelar as tarefas dos demais). Para finalizar o exemplo, esse cluster possui 8 computadores do mesmo modelo, sistema operacional Linux e o programa foi feito em C, com uma biblioteca MPI (para programação paralela).

Importante levar para o prova que um cluster é um sistema fortemente acoplado, ou seja, os computadores compartilham uma rede local, o mesmo sistema operacional em cada computador e há troca de mensagens entre eles. Note que foge um pouco do conceito geral de sistemas distribuídos, ficando mais voltado para um sistema paralelo. Mas para efeito de concurso, os clusters são cobrados junto com os grids (que veremos na sequência).

A computação em grid (grade – em português) possui uma estrutura de processamento descentralizada que não exige um tipo de controle central (não há um nó mestre), trabalhando em um ambiente cooperativo. Possui uma estrutura fracamente acoplada, com múltiplas máquinas interligadas através de uma rede de computadores. Por exemplo, milhares de máquinas no mundo, com sistemas operacionais diferentes, utilizadas para resolver cálculos para encontrar vida extraterrestre (já vimos nesta aula). Não há um requisito de alta disponibilidade dos equipamentos e a localização dos equipamentos é transparente para os usuários.



1. (CESPE/IPHAN - 2018) Cluster e GRID possuem características semelhantes, como o multiprocessamento distribuído, mas se distinguem pelas seguintes razões: cluster é uma coleção de máquinas paralelas conectadas entre si que, por trabalharem juntas, fornecem alta disponibilidade e(ou) balanceamento de carga; o GRID, por sua vez, envolve a integração, gestão de recursos computacionais fracamente acoplados e geograficamente distribuídos.

Comentários:

Um belo resumo! Para o cluster pense em máquinas em uma mesma rede, numa mesma "sala", são fortemente acoplados (mesmo sistema operacional, mesmo tipo de hardware). Os grids são espalhados pelo mundo, fracamente acoplados (sistemas operacionais e máquinas diferentes). Portanto, a questão está correta.

Gabarito: Correta

2. (CESPE/TRE-PE - 2015) Os sistemas operacionais têm rotinas que não são executadas de forma linear, mas, sim, concorrentemente, em função de eventos assíncronos.

Comentários:

As rotinas do sistema operacional são executadas conforme as características das tarefas (jobs). Existem os jobs do tipo CPU bound e os I/O bound. Os eventos variam conforme a necessidade dos usuários (mais uso do processador ou de dispositivos de entrada/saída). Por isso, a assertiva afirmou que as rotinas são executadas em função de eventos assíncronos! Portanto, a questão está correta.

Gabarito: Correta

QUESTÕES COMENTADAS - SISTEMAS DISTRIBUÍDOS - MULTIBANCAS

1. (CESPE/MBASA - 2010) Clusters ou combinações de clusters são usados quando os conteúdos são críticos, apesar de não haver necessidade de estarem disponíveis e (ou) processados rapidamente.

Comentários:

Os clusters são estruturas cujo propósito é proporcionar alta disponibilidade, em situações nas quais há requisitos de criticidade de fornecimento do conteúdo, portanto há a necessidade que estejam disponíveis e geralmente se deseja um processamento rápido. Portanto, a questão está errada.

Gabarito: Errada

2. (IADES/PG-DF - 2011) Segundo Andrew Tanembaum (2007) "Sistema Distribuído é uma coleção de computadores independentes que se apresenta ao usuário como um sistema único e consistente". Assinale a alternativa correta a respeito de um sistema de informação distribuído.

A) A distribuição de tarefas se dá a partir de requisições do usuário, que indica o endereço do servidor onde deseja executar tal tarefa.

B) Em uma rede de computadores há servidores dedicados a atender pedidos dos clientes e estes, por sua vez, têm função exclusiva de requisitantes.

C) Todos os computadores de uma rede executam tarefas de cliente e servidor, quando se deseja integrá-los em uma arquitetura de sistemas distribuídos.

D) A transparência de acesso é uma característica dos sistemas distribuídos que permite que recursos sejam acessados sem que sua localização seja determinada.

E) Em um sistema de objetos distribuídos é possível invocar métodos de um objeto, ainda que este não esteja presente no computador do usuário.

Comentários:

(A) A distribuição de tarefas indica a tarefa a ser desempenhada, sem indicar o endereço do servidor onde será executada tal tarefa. Lembre que um dos princípios de sistemas distribuídos é a transparência, portanto o usuário não tem ideia qual servidor será utilizado.

(B) Em um sistema distribuído não há a figura fixa dos servidores dedicados a atender requisições.

(C) Todos? Se for utilizada a arquitetura peer-to-peer até pode ser, mas quando é utilizada a arquitetura cliente-servidor?

(D) O conceito mostrado é o da transparência de localização, não de acesso! Vejamos:

- Transparência de acesso: oculta diferenças na representação de dados e no modo de acesso a um recurso.
- Transparência de localização: oculta o lugar em que um recurso está localizado.

(E) É isso aí! Em um sistema de objetos distribuídos é possível invocar métodos de um objeto remoto. Para tanto, podemos fazer uso de tecnologias como RMI (Remote Method Invocation) ou Corba.

Portanto, a alternativa E está correta e é o gabarito da questão.

Gabarito: Letra E

3. (FCC/TRE-PE - 2011) Arquitetura padrão proposta pelo Object Management Group (OMG) para estabelecer e simplificar a troca de dados entre sistemas distribuídos heterogêneos por meio de uma estrutura comum para o gerenciamento de objetos distribuídos que é conhecida como Object Manager Architecture (OMA). Trata-se de

A) IDL.

B) RPC.

C) DCON.

D) CORBA.

E) COM.

Comentários:

Uma das possibilidades de invocação de métodos remotos em sistemas distribuídos é através do uso de Corba ou RMI. RMI é mais utilizado, por ser mais simples que Corba. Corba é a arquitetura padrão proposta pelo Object Management Group (OMG) para estabelecer e simplificar a troca de dados entre sistemas distribuídos heterogêneos por meio de uma estrutura comum para o gerenciamento de objetos distribuídos. Portanto, a alternativa D está correta e é o gabarito da questão.

Gabarito: Letra D

4. (CESPE/SERPRO - 2013) Com relação à arquitetura de sistemas distribuídos, julgue os próximos itens. Na arquitetura distribuída, os sistemas orientados a eventos possuem processos fortemente acoplados.

Comentários:

Na arquitetura distribuída, os sistemas orientados a eventos possuem processos fracamente acoplados. Portanto, a questão está errada.

Gabarito: Errada

5. (FUNCAB/MDA - 2014) São desvantagens dos sistemas distribuídos:

A) dependência de hardware e custo maior de desenvolvimento.

B) custo maior de desenvolvimento e maior probabilidade de ocorrência de falhas (bugs) no programa.

C) dependência de sistemas operacionais e necessidade de maior processamento.

D) necessidade de maior processamento e pouca disponibilidade.

E) ausência de controle das suas operações e dependência do tipo de rede utilizada para ligar as localidades.

Comentários:

(A) Sistemas distribuídos são heterogêneos (plataformas de hardware e sistemas operacionais distintos). (B) Sistemas distribuídos possuem maior complexidade de desenvolvimento (mais chance de bugs) e maior custo. (C) Sistemas distribuídos não dependem de sistemas operacionais específicos. (D) Sistemas distribuídos estão mais disponíveis. (E) Sistemas distribuídos permitem controle das operações e não dependem do tipo de rede utilizada para ligar as localidades. Portanto, a alternativa B está correta e é o gabarito da questão.

Gabarito: Letra B

6. (CESPE/TRE-PE - 2015) Os sistemas operacionais têm rotinas que não são executadas de forma linear, mas, sim, concorrentemente, em função de eventos assíncronos.

Comentários:

As rotinas do sistema operacional são executadas conforme as características das tarefas (jobs). Existem os jobs do tipo CPU bound e os I/O bound. Os eventos variam conforme a necessidade dos usuários (mais uso do processador ou de dispositivos de entrada/saída). Por isso, a assertiva

afirmou que as rotinas são executadas em função de eventos assíncronos! Portanto, a questão está correta.

Gabarito: Correta

7. (CESPE/TJ-DFT - 2015) Nos sistemas implementados a partir do uso de uma arquitetura de componentes distribuídos, o middleware tem a responsabilidade de gerenciar a interação entre esses componentes.

Comentários:

Middleware: camada de software localizada logicamente entre uma camada de nível mais alto (usuários e aplicações) e uma camada subjacente (frameworks e facilidades de comunicação). As principais funções da camada de middleware são:

- Ocultar do usuário que a aplicação é executada em diferentes máquinas distribuídas geograficamente;
- Ocultar a heterogeneidade dos sistemas operacionais (ex.: Windows, Linux etc.) e protocolos de comunicação subjacentes.

Portanto, a questão está correta.

Gabarito: Correta

8. (CESPE/STJ - 2015) A arquitetura orientada a serviços é forma de desenvolvimento de sistemas distribuídos em que os componentes de sistemas são serviços autônomos, razão por que, devido à interoperabilidade, as ligações entre os serviços devem ser rígidas para não provocar mudanças durante sua execução.

Comentários:

Não é o foco de nossa aula a arquitetura orientada a serviços. Mas como é um assunto pertinente, vamos ver... A arquitetura orientada a serviços é forma de desenvolvimento de sistemas distribuídos em que os componentes são autônomos e as ligações entre os serviços devem ser flexíveis durante sua execução. Portanto, a questão está errada.

Gabarito: Errada

9. (IBFC/EBSERH - 2017) Assinale a alternativa correta. Cluster é um conceito que está diretamente relacionado aos sistemas de alta disponibilidade. Existem vários tipos de cluster, no entanto há alguns que são mais conhecidos, como:

A) cluster paralelo - cluster de alta disponibilidade - cluster de três camadas

B) cluster de alto desempenho - cluster virtual - cluster matricial

C) cluster paralelo - cluster virtual - cluster para balanceamento de carga

D) cluster de alto desempenho - cluster matricial - cluster de três camadas

E) cluster de alto desempenho - cluster de alta disponibilidade - cluster para balanceamento de carga

Comentários:

Clusters são sistemas de processamento compostos por uma coleção de computadores autônomos, interconectados e que trabalham em conjunto para processar uma tarefa. Tipicamente na computação de cluster é utilizada programação paralela na qual um único programa é executado em paralelo. Dependendo do objetivo que se pretenda, podemos ter os seguintes tipos de cluster:

- Cluster de alto desempenho: direcionado a aplicações bastante exigentes no que diz respeito ao processamento;
- Cluster de alta disponibilidade: tem como objetivo permanecer ativo por um longo período de tempo e em plena condição de uso. Consegue detectar erros e se proteger de possíveis falhas;
- Cluster de balanceamento de carga: controla a distribuição equilibrada do processamento. As tarefas de processamento são distribuídas o mais uniformemente possível entre os nós.

Portanto, a alternativa E está correta e é o gabarito da questão.

Gabarito: Letra E

10. (FGV/IBGE - 2017) Keyse é gerente de um centro de dados que hospeda distintos tipos de aplicação. Recentemente Keyse recebeu uma solicitação de hospedagem de uma aplicação de missão crítica, a qual requer uma disponibilidade de 99,8% ao ano.

Considerando V para as afirmativas verdadeiras e F para as falsas, analise as características do cluster para atender o requisito de alta disponibilidade do sistema.

( ) capacidade de distribuir igualmente todo o tráfego de entrada entre todos os nós do cluster, para evitar a sobrecarga de requisições a qualquer um deles e o consequente travamento do cluster;

( ) ter nós em espera, em quantidade suficiente, para assumir automaticamente a função de outro nó defeituoso;

( ) realizar processamento da aplicação de forma paralela entre os vários nós do cluster;

( ) monitoramento dos nós feito por eles mesmos, por uma rede diferente da rede de dados.

A sequência correta é:

A) V – F – V – F;

B) F – V – F – V;

C) V – V – V – F;

D) V – V – F – F;

E) F – F – V – V.

Comentários:

O foco do cluster é a alta disponibilidade, então vejamos:

(F) Seria verdadeira se o foco fosse o balanceamento de carga!

(V) Ter nós sobrando é importante para garantir a alta disponibilidade, no caso de falhas de outros

(F) O foco é manter o funcionamento (alta disponibilidade) e não paralelizar a execução dos programas (o que seria o foco do alto desempenho).

(V) O monitoramento dos nós feito por eles mesmos, sem depender de "alguém de fora" e a rede utilizada é separada da rede de dados.

Portanto, a alternativa B está correta e é o gabarito da questão.

Gabarito: Letra B

11. (CESPE/IPHAN - 2018) Cluster e GRID possuem características semelhantes, como o multiprocessamento distribuído, mas se distinguem pelas seguintes razões: cluster é uma coleção de máquinas paralelas conectadas entre si que, por trabalharem juntas, fornecem alta disponibilidade e(ou) balanceamento de carga; o GRID, por sua vez, envolve a integração, gestão de recursos computacionais fracamente acoplados e geograficamente distribuídos.

Comentários:

Um belo resumo! Para o cluster pense em máquinas em uma mesma rede, numa mesma "sala", são fortemente acoplados (mesmo sistema operacional, mesmo tipo de hardware). Os grids são espalhados pelo mundo, fracamente acoplados (sistemas operacionais e máquinas diferentes). Portanto, a questão está correta.

Gabarito: Correta

12. (UFLA/UFLA - 2018) O aumento da facilidade de acesso à Internet tem permitido uma grande disponibilização da informação. Para dar suporte a essa facilidade de acesso é necessária uma enorme infraestrutura de hardware e software. Considerando as características de computadores paralelos, analise as proposições a seguir:

I. Um sistema de multiprocessamento simétrico pode ser composto por milhares de computadores com processadores e sistemas operacionais heterogêneos.

II. Um sistema de processamento paralelo em massa visa resolver problemas que exigem capacidade de utilização de memória compartilhada usando um único conjunto de núcleos de processamento localizados em um mesmo computador.

III. Um cluster de computadores é uma coleção de dois ou mais computadores usados para executar um dado problema podendo conter processadores multicore.

IV. Um grid computing provê uma plataforma na qual recursos computacionais são organizados dentro de um ou mais conjuntos lógicos, as tarefas são divididas entre diversos computadores locais ou remotos formando um "super computador virtual".

Assinale a alternativa CORRETA:

A) Somente as proposições I e II estão corretas.

B) Somente as proposições II e III estão corretas.

C) Somente as proposições III e IV estão corretas.

D) Somente as proposições I e IV estão corretas.

Comentários:

(I) Não vimos nesta aula, mas não tem como garantir a simetria com milhares de computadores com processadores e sistemas operacionais distintos!

(II) Um sistema de processamento paralelo (cluster) utiliza troca de mensagens, pois cada computador tem sua memória. Ex.: o mestre envia tarefas aos escravos (via troca de mensagens), os escravos realizam as tarefas e retornam com os resultados. E por aí vai...

(III) Exato, um agregado de computadores (multicore ou não), utilizados para executar um dado problema (ex.: quebrar uma senha por força bruta).

(IV) Grid é fracamente acoplado, geralmente "espalhado pelo mundo", tendo os recursos computacionais organizados dentro de um ou mais conjuntos lógicos. A ideia é possuir um "super computador virtual".

Portanto, a alternativa C está correta e é o gabarito da questão.

Gabarito: Letra C

13. (INSTITUTO AOCP/PC-ES - 2019) Se um servidor Web estiver sobrecarregado, basta agregar novos servidores e dividir a carga de processamento. Considerando o funcionamento dos principais serviços de rede, como é conhecido o esquema de agregação de servidores?

A) SIP Apache.

B) Divserver.

C) Cluster.

D) Upload.

E) ARPANET.

Comentários:

Clusters são sistemas de processamento compostos por uma coleção de computadores autônomos, interconectados e que trabalham em conjunto para processar uma tarefa. Tipicamente na computação de cluster é utilizada programação paralela na qual um único programa é executado em paralelo. Dependendo do objetivo que se pretenda, podemos ter os seguintes tipos de cluster:

- Cluster de alto desempenho: direcionado a aplicações bastante exigentes no que diz respeito ao processamento;
- Cluster de alta disponibilidade: tem como objetivo permanecer ativo por um longo período e em plena condição de uso. Consegue detectar erros e se proteger de possíveis falhas;
- Cluster de balanceamento de carga: controla a distribuição equilibrada do processamento. As tarefas de processamento são distribuídas o mais uniformemente possível entre os nós.

Portanto, a alternativa C está correta e é o gabarito da questão.

Gabarito: Letra C

14. (UFMG/UFMG - 2019) Um Cluster de Alta Disponibilidade é caracterizado por

A) no máximo 2 servidores para redundância ou tolerância a falhas e fontes de energia redundantes, dentre outros dispositivos.

B) diversos servidores para redundância ou tolerância a falhas e fontes de energia redundantes, dentre outros dispositivos.

C) diversos servidores conectados por uma rede wifi e fontes de energia redundantes, dentre outros dispositivos.

D) um único servidor com diversas fontes de energia redundantes ligados a um gerador para que esse servidor quase nunca desligue por falta de energia.

Comentários:

Cluster de alta disponibilidade: tem como objetivo permanecer ativo por um longo período e em plena condição de uso. Consegue detectar erros e se proteger de possíveis falhas.

Ou seja, deve haver diversos servidores e fonte de energia redundante (geradores, fornecimento de energia por mais de uma empresa etc.). Portanto, a **alternativa B** está correta e é o gabarito da questão.

Gabarito: Letra B

LISTA DE QUESTÕES - SISTEMAS DISTRIBUÍDOS - MULTIBANCAS

1. (CESPE/MBASA - 2010) Clusters ou combinações de clusters são usados quando os conteúdos são críticos, apesar de não haver necessidade de estarem disponíveis e (ou) processados rapidamente.

2. (IADES/PG-DF - 2011) Segundo Andrew Tanembaum (2007) "Sistema Distribuído é uma coleção de computadores independentes que se apresenta ao usuário como um sistema único e consistente". Assinale a alternativa correta a respeito de um sistema de informação distribuído.

A) A distribuição de tarefas se dá a partir de requisições do usuário, que indica o endereço do servidor onde deseja executar tal tarefa.

B) Em uma rede de computadores há servidores dedicados a atender pedidos dos clientes e estes, por sua vez, têm função exclusiva de requisitantes.

C) Todos os computadores de uma rede executam tarefas de cliente e servidor, quando se deseja integrá-los em uma arquitetura de sistemas distribuídos.

D) A transparência de acesso é uma característica dos sistemas distribuídos que permite que recursos sejam acessados sem que sua localização seja determinada.

E) Em um sistema de objetos distribuídos é possível invocar métodos de um objeto, ainda que este não esteja presente no computador do usuário.

3. (FCC/TRE-PE - 2011) Arquitetura padrão proposta pelo Object Management Group (OMG) para estabelecer e simplificar a troca de dados entre sistemas distribuídos heterogêneos por meio de uma estrutura comum para o gerenciamento de objetos distribuídos que é conhecida como Object Manager Architecture (OMA). Trata-se de

A) IDL.

B) RPC.

C) DCON.

D) CORBA.

E) COM.

4. (CESPE/SERPRO - 2013) Com relação à arquitetura de sistemas distribuídos, julgue os próximos itens. Na arquitetura distribuída, os sistemas orientados a eventos possuem processos fortemente acoplados.
5. (FUNCAB/MDA - 2014) São desvantagens dos sistemas distribuídos:

A) dependência de hardware e custo maior de desenvolvimento.

B) custo maior de desenvolvimento e maior probabilidade de ocorrência de falhas (bugs) no programa.

C) dependência de sistemas operacionais e necessidade de maior processamento.

D) necessidade de maior processamento e pouca disponibilidade.

E) ausência de controle das suas operações e dependência do tipo de rede utilizada para ligar as localidades.

6. (CESPE/TRE-PE - 2015) Os sistemas operacionais têm rotinas que não são executadas de forma linear, mas, sim, concorrentemente, em função de eventos assíncronos.
7. (CESPE/TJ-DFT - 2015) Nos sistemas implementados a partir do uso de uma arquitetura de componentes distribuídos, o middleware tem a responsabilidade de gerenciar a interação entre esses componentes.
8. (CESPE/STJ - 2015) A arquitetura orientada a serviços é forma de desenvolvimento de sistemas distribuídos em que os componentes de sistemas são serviços autônomos, razão por que, devido à interoperabilidade, as ligações entre os serviços devem ser rígidas para não provocar mudanças durante sua execução.
9. (IBFC/EBSERH - 2017) Assinale a alternativa correta. Cluster é um conceito que está diretamente relacionado aos sistemas de alta disponibilidade. Existem vários tipos de cluster, no entanto há alguns que são mais conhecidos, como:

A) cluster paralelo - cluster de alta disponibilidade - cluster de três camadas

B) cluster de alto desempenho - cluster virtual - cluster matricial

C) cluster paralelo - cluster virtual - cluster para balanceamento de carga

D) cluster de alto desempenho - cluster matricial - cluster de três camadas

E) cluster de alto desempenho - cluster de alta disponibilidade - cluster para balanceamento de carga

10. (FGV/IBGE - 2017) Keyse é gerente de um centro de dados que hospeda distintos tipos de aplicação. Recentemente Keyse recebeu uma solicitação de hospedagem de uma aplicação de missão crítica, a qual requer uma disponibilidade de 99,8% ao ano.

Considerando V para as afirmativas verdadeiras e F para as falsas, analise as características do cluster para atender o requisito de alta disponibilidade do sistema.

( ) capacidade de distribuir igualmente todo o tráfego de entrada entre todos os nós do cluster, para evitar a sobrecarga de requisições a qualquer um deles e o consequente travamento do cluster;

( ) ter nós em espera, em quantidade suficiente, para assumir automaticamente a função de outro nó defeituoso;

( ) realizar processamento da aplicação de forma paralela entre os vários nós do cluster;

( ) monitoramento dos nós feito por eles mesmos, por uma rede diferente da rede de dados.

A sequência correta é:

A) V – F – V – F;

B) F – V – F – V;

C) V – V – V – F;

D) V – V – F – F;

E) F – F – V – V.

11. (CESPE/IPHAN - 2018) Cluster e GRID possuem características semelhantes, como o multiprocessamento distribuído, mas se distinguem pelas seguintes razões: cluster é uma coleção de máquinas paralelas conectadas entre si que, por trabalharem juntas, fornecem alta disponibilidade e(ou) balanceamento de carga; o GRID, por sua vez, envolve a integração, gestão de recursos computacionais fracamente acoplados e geograficamente distribuídos.

12. (UFLA/UFLA - 2018) O aumento da facilidade de acesso à Internet tem permitido uma grande disponibilização da informação. Para dar suporte a essa facilidade de acesso é necessária uma enorme infraestrutura de hardware e software. Considerando as características de computadores paralelos, analise as proposições a seguir:

I. Um sistema de multiprocessamento simétrico pode ser composto por milhares de computadores com processadores e sistemas operacionais heterogêneos.

II. Um sistema de processamento paralelo em massa visa resolver problemas que exigem capacidade de utilização de memória compartilhada usando um único conjunto de núcleos de processamento localizados em um mesmo computador.

III. Um cluster de computadores é uma coleção de dois ou mais computadores usados para executar um dado problema podendo conter processadores multicore.

IV. Um grid computing provê uma plataforma na qual recursos computacionais são organizados dentro de um ou mais conjuntos lógicos, as tarefas são divididas entre diversos computadores locais ou remotos formando um "super computador virtual".

Assinale a alternativa CORRETA:

A) Somente as proposições I e II estão corretas.

B) Somente as proposições II e III estão corretas.

C) Somente as proposições III e IV estão corretas.

D) Somente as proposições I e IV estão corretas.

13. (INSTITUTO AOCP/PC-ES - 2019) Se um servidor Web estiver sobrecarregado, basta agregar novos servidores e dividir a carga de processamento. Considerando o funcionamento dos principais serviços de rede, como é conhecido o esquema de agregação de servidores?

A) SIP Apache.

B) Divserver.

C) Cluster.

D) Upload.

E) ARPANET.

14. (UFMG/UFMG - 2019) Um Cluster de Alta Disponibilidade é caracterizado por

A) no máximo 2 servidores para redundância ou tolerância a falhas e fontes de energia redundantes, dentre outros dispositivos.

B) diversos servidores para redundância ou tolerância a falhas e fontes de energia redundantes, dentre outros dispositivos.

C) diversos servidores conectados por uma rede wifi e fontes de energia redundantes, dentre outros dispositivos.

D) um único servidor com diversas fontes de energia redundantes ligados a um gerador para que esse servidor quase nunca desligue por falta de energia.

GABARITO

1- Errada
2- E
3- D
4- Errada
5- B
6- Correta
7- Correta
8- Errada
9- E
10- B
11- Correta
12- C
13- C
14- B

Resiliência de Aplicações

Um assunto que tem muito a ver com sistemas distribuídos é a resiliência de aplicações, que se refere à capacidade de um sistema continuar em funcionamento de maneira satisfatória, fornecendo serviços mesmo quando ocorrem falhas ou eventos atípicos. Isso envolve a capacidade de se recuperar automaticamente de falhas, lidar com altos volumes de tráfego e manter um desempenho aceitável sob condições adversas.

Alguns princípios e práticas comumente utilizados para aumentar a resiliência de aplicações são:

- Projeto para eventuais falhas: a ocorrência de possíveis falhas devem ser assumidas, então deve-se projetar o sistema para lidar com elas de forma elegante, sem interrupções significativas;
- Redundância: envolve a replicação de componentes críticos (servidores, bancos de dados, entre outros) para que haja backups disponíveis em caso de falha;
- Gestão de capacidade: envolve o monitoramento e ajuste dinâmico dos recursos de acordo com a necessidade de lidar com picos de tráfego ou demandas inesperadas;
- Isolamento de falhas: envolve o projeto de componentes do sistema de forma que uma falha em um componente não afete os outros. Para isso, utiliza-se contêineres, microsserviços ou outras técnicas similares;
- Respostas graciosas: envolve a implementação de mecanismos de fallback (veremos adiante) e degradação graciosa para garantir que partes essenciais do sistema possam continuar em funcionamento mesmo quando serviços secundários falharem;
- Testes de resiliência: envolve a realização de testes de estresse e de recuperação para identificar vulnerabilidades, com o intuito de melhorar a capacidade do sistema de lidar com situações adversas.

Cache

No contexto de resiliência de aplicações, o uso de cache pode desempenhar um papel fundamental no tocante à melhoria de desempenho e à capacidade de resposta do sistema, além de contribuir para a resiliência de várias maneiras:

- Redução da carga em recursos críticos: Com a utilização de cache para armazenar dados frequentemente acessados, há uma tendência de redução da carga em sistemas de armazenamento principal, como por exemplo, bancos de dados. Isso, além de melhorar o desempenho geral, também protege contra falhas ou picos de tráfego que poderiam sobrecarregar tais recursos;
- Aumento da tolerância a falhas: Se houver falha de um serviço ou recurso externo, um cache bem projetado pode continuar a servir dados que foram previamente armazenados. Com isso, partes da aplicação podem manter o funcionamento, enquanto a falha é

resolvida;

- Melhoria da disponibilidade e latência: Ao reduzir a dependência de serviços externos e ao armazenar dados localmente em cache, as aplicações podem reduzir a latência percebida pelos usuários finais. Com isso, mesmo quando ocorrer uma falha parcial, o sistema pode continuar fornecendo respostas rápidas;

- Capacidade de suportar picos de tráfego: Durante picos de tráfego, um cache eficiente pode ajudar a absorver parte da carga extra, realizando uma distribuição entre os dados que já estiverem em cache e reduzindo a necessidade de acesso direto a recursos mais lentos;

- Consistência e atualização de dados: Embora o cache melhore o desempenho e a resiliência, é importante considerar estratégias para garantir que os dados em cache sejam atualizados regularmente e consistentemente. Para isso, existem técnicas como expiração de cache, invalidação de cache e atualizações assíncronas.

Fallback

Fallback significa "retroceder". No contexto de resiliência de aplicações, *fallback* refere-se a uma estratégia ou mecanismo alternativo que uma aplicação pode adotar quando ocorre uma falha ou quando um recurso esperado não está disponível. É uma técnica fundamental para garantir que a aplicação possa continuar a funcionar de maneira aceitável, mesmo diante de situações adversas. Algumas características sobre o uso de *fallback* na resiliência de aplicações são:

- Substituição de funcionalidade: Quando ocorre uma falha de um serviço ou recurso principal, o *fallback* permite que a aplicação substitua essa funcionalidade por uma alternativa. Por exemplo, se um serviço de reservas estiver fora do ar, a aplicação pode implementar um *fallback* para usar um provedor de reservas secundário;

- Degradação graciosa: O *fallback* pode ser usado para permitir uma degradação controlada do serviço, o que é melhor do que uma falha completa! Por exemplo, se um recurso para determinada funcionalidade específica não estiver disponível, a aplicação pode substituí-lo por uma versão mais simples ou uma mensagem de erro amigável, garantindo que partes essenciais da aplicação continuem em funcionamento;

- Desempenho e latência: A implementação de *fallback* pode ajudar a melhorar o desempenho e reduzir a latência percebida pelos usuários. Ao invés de esperar um longo período por uma resposta de um serviço lento ou indisponível, a aplicação pode rapidamente recorrer a uma alternativa que ofereça uma resposta mais rápida;

- Tolerância a falhas: Com o planejamento de estratégias de *fallback*, as aplicações podem aumentar sua tolerância a falhas, ou seja, mesmo quando ocorrerem problemas técnicos ou interrupções de serviços, a aplicação pode continuar operando de maneira limitada ou com funcionalidades essenciais, com a continuidade do serviço;

- Monitoramento e gestão: Não adianta ter a implementação de *fallback*, se não houver o monitoramento e gerenciamento para garantir que eles sejam ativados apenas quando necessário e que não comprometam a integridade dos dados ou a segurança da aplicação. Por isso, as estratégias de *fallback* devem ser testadas regularmente.

Circuitbrake

No contexto de resiliência de aplicações, *circuit breaker* é um padrão de projeto utilizado para lidar com falhas em sistemas distribuídos ou serviços externos que uma aplicação depende. Tal padrão é inspirado em conceitos elétricos, onde um disjuntor (*circuit breaker*) interrompe o fluxo de eletricidade em caso de sobrecarga ou curto-circuito para proteger o sistema elétrico. Em relação à resiliência, o *circuit breaker* possui as seguintes características:

- Monitoramento de falhas: Há um monitoramento das chamadas ou requisições feitas a um serviço externo. Se uma quantidade limite de falhas ocorrer dentro de um período definido, o *circuit breaker* muda para um estado "aberto";

- Abertura do circuito: Quando o *circuit breaker* está aberto, há um impedimento de chamadas adicionais ao serviço externo. Em vez disso, a aplicação pode retornar rapidamente com uma resposta alternativa (como um *fallback*, por exemplo) ou uma mensagem de erro amigável;

- Retentativa: Após um período de tempo (*timeout*), o *circuit breaker* pode permitir que uma única requisição seja realizada ao serviço externo para verificar se ele está novamente disponível. Se a resposta for positiva, o *circuit breaker* pode ser redefinido para um estado fechado e voltar a permitir chamadas normais.

Disaster Recovery

No contexto de resiliência de aplicações, *Disaster Recovery* (Recuperação de Desastres) refere-se ao conjunto de políticas, procedimentos e tecnologias implementadas para restaurar rapidamente uma aplicação ou sistema após um incidente grave que cause uma interrupção significativa ou a incapacidade de operação normal. Os objetivos do *Disaster Recovery* são:

- Minimização de Tempo de Inatividade: Reduzir ao máximo o tempo necessário para restaurar a funcionalidade completa da aplicação após um desastre;

- Manutenção da Integridade dos Dados: Garantir que todos os dados críticos estejam protegidos e disponíveis após a recuperação;

- Continuidade Operacional: Assegurar que a aplicação possa retomar suas operações normais o mais rápido possível, mesmo em circunstâncias adversas.

Os componentes principais do *Disaster Recovery* são:

- *Backup* de Dados: Cópias (*backups*) regulares e seguras dos dados críticos da aplicação são fundamentais. Podem ser utilizadas as estratégias de *backup* incremental, diferencial e

completo, com o armazenamento local e também geograficamente separado;

- Replicação de Dados: Envolve a manutenção de cópias em tempo real (ou quase real) dos dados em diferentes locais físicos, ou na nuvem, para garantir a disponibilidade contínua;

- Planos de Continuidade de Negócios: Envolve a documentação detalhada que define como a aplicação será restaurada, incluindo quais sistemas e dados prioritários devem ser recuperados primeiro e quais procedimentos devem ser seguidos;

- Testes e Simulações: Testes de recuperação de desastres devem ser realizados regularmente para garantir que os planos sejam eficazes e que a equipe esteja familiarizada com os procedimentos de recuperação;

- Redundância de Infraestrutura: Envolve a implementação de sistemas redundantes, como por exemplo, servidores e redes. Se a infraestrutura principal falhar, os sistemas redundantes podem entrar em operação rapidamente.

Contingência

No contexto de resiliência de aplicações, contingência se refere à preparação e resposta planejada para situações adversas ou eventos inesperados. Essas situações ou eventos podem impactar a disponibilidade, desempenho ou funcionalidade da aplicação. São aspectos importantes da contingência:

- Identificação de Riscos: Envolve a compreensão e avaliação dos potenciais riscos que podem afetar a aplicação. Exemplos: falhas de hardware, interrupções de rede, ataques cibernéticos, desastres naturais etc.;

- Planejamento de Resposta: Envolve o desenvolvimento de procedimentos e protocolos claros para responder rapidamente a situações de emergência, o que inclui a definição de papéis e responsabilidades da equipe durante uma crise;

- Redundância e Backup: Implementação de sistemas redundantes e backups regulares para garantir a disponibilidade contínua de dados e funcionalidades críticas da aplicação;

- Testes de Contingência: Envolve a realização periódica de simulações e testes de contingência para garantir que os planos de resposta funcionem efetivamente e que a equipe esteja familiarizada para lidar com emergências reais;

- Monitoramento e Alertas: Envolve a utilização de ferramentas e sistemas de monitoramento contínuo para detectar problemas potenciais precocemente e acionar alertas para que ações corretivas possam ser tomadas rapidamente.

Balanceamento de Carga Global de Servidores (GSLB)

O Balanceamento de Carga Global de Servidores (GSLB - *Global Server Load Balancing*) é uma técnica avançada utilizada para distribuir o tráfego de rede (balanceamento) de forma eficiente

entre múltiplos servidores ou *data centers* geograficamente distribuídos. Essa abordagem visa melhorar o desempenho e a disponibilidade, além de aprimorar a resiliência e a escalabilidade das aplicações e serviços. As funcionalidades e princípios do GSLB envolvem:

- Redução da Latência: O GSLB pode direcionar os usuários para o servidor mais próximo geograficamente, fazendo com que a latência seja reduzida;

- Balanceamento de Carga Dinâmico: O GSLB pode levar em consideração a carga atual dos servidores, a qualidade da rede e outros parâmetros dinâmicos para tomar decisões de roteamento;

- Resiliência e Alta Disponibilidade: Quando ocorre uma falha em um servidor ou *data center*, o GSLB pode redirecionar automaticamente o tráfego para outros servidores ou locais que estejam operacionais, garantindo a continuidade do serviço;

- Geodiversidade: Permite que aplicações globais sejam distribuídas entre múltiplas localizações geográficas. Com isso, há uma melhoria na resiliência contra desastres naturais, falhas regionais de rede ou problemas políticos que possam afetar a conectividade em determinada região;

- Monitoramento e Saúde do Servidor: O GSLB frequentemente inclui recursos de monitoramento de saúde de servidor, com a verificação da disponibilidade e a capacidade dos servidores de lidar com o tráfego antes de direcionar novas solicitações para eles.

Site Ativo x Ativo

Um "Site Ativo x Ativo" (*Active-Active Site*) é uma arquitetura de alta disponibilidade em que múltiplos *sites* ou *data centers* estão totalmente operacionais e atendendo ao tráfego simultaneamente. Essa abordagem é projetada para melhorar a resiliência e a escalabilidade de aplicações e serviços online. Desta forma, o tráfego é distribuído de forma equilibrada entre esses diferentes locais. As características principais do modelo *Active-Active* são:

- Redundância Geográfica: Consiste em ter múltiplos locais geograficamente distribuídos, cada um capaz de lidar com a carga total de trabalho. Com isso, há uma redução no risco de uma única falha de *data center* ou região afetar a disponibilidade do serviço;

- Distribuição de Carga: O tráfego é distribuído de forma balanceada entre os diferentes *sites* ativos. Isso pode ser realizado através de técnicas como balanceamento de carga global (GSLB), para garantir que cada *site* receba uma carga proporcional ao seu poder de processamento e capacidade de rede;

- Resiliência a Falhas: Em caso de falha em um dos *sites* ou *data centers*, o tráfego pode ser automaticamente redirecionado para os *sites* restantes que ainda estão operacionais. Isso garante a continuidade do serviço e minimiza o impacto percebido pelos usuários finais;

- Escalabilidade Horizontal: A adição de novos *sites* ativos permite escalar horizontalmente a capacidade de processamento e a largura de banda disponível, conforme a demanda

aumenta;

- Atualizações e Manutenção: A arquitetura *Active-Active* facilita a realização de atualizações e manutenções sem interromper o serviço globalmente. Isso pode ser alcançado através da desativação de um *site* de cada vez, enquanto os outros continuam com o funcionamento normal.

Failover

Failover é uma solução redundante considerada sofisticada. Quando o equipamento principal falha, o equipamento redundante continua a responder a requisição, sem interromper a comunicação, sendo que para isso há a replicação de estados, ou seja as informações são replicadas/compartilhadas entre os dois equipamentos.

Quando o componente principal falha, o sistema automaticamente transfere suas operações para um componente secundário ou de backup, garantindo a continuidade do serviço. Esse processo é comum em sistemas que exigem alta disponibilidade, como servidores, bancos de dados e redes. Os dois tipos principais de failover são:

- Automático: O sistema detecta a falha e realiza a transferência de forma automática, sem intervenção manual;

- Manual: Um operador precisa intervir para realizar a troca para o sistema de backup.

Por exemplo, em um ambiente crítico, como um hospital, se um mecanismo de failover for implementado em um servidor de banco de dados, se o servidor principal falhar, um servidor secundário (de backup) assume as operações para evitar interrupções.

Questões Comentadas - Resiliência - CEBRASPE

1. (CEBRASPE/SERPRO/2023) Quanto à resiliência de aplicações, referente à capacidade de um aplicativo de fornecer seus serviços normalmente mesmo diante de situações adversas, julgue o item a seguir.

   Arquiteturas de site ativo × ativo são utilizadas para garantir a continuidade dos negócios e fornecer aos usuários uma experiência ininterrupta.

Comentários:

Um "Site Ativo x Ativo" (*Active-Active Site*) é uma arquitetura de alta disponibilidade em que múltiplos *sites* ou *data centers* estão totalmente operacionais e atendendo ao tráfego simultaneamente. Essa abordagem é projetada para melhorar a resiliência e a escalabilidade de aplicações e serviços online. Desta forma, o tráfego é distribuído de forma equilibrada entre esses diferentes locais.

Gabarito: Certo

2. (CEBRASPE/MPO/2024) Julgue o item a seguir, relativo ao teste de estresse, a testes automatizados e à resiliência de aplicações.

   O uso de cache e a implementação de mecanismos de fallback são estratégias importantes para aumentar a resiliência de uma aplicação.

Comentários:

No contexto de resiliência de aplicações, o uso de cache pode desempenhar um papel fundamental no tocante à melhoria de desempenho e à capacidade de resposta do sistema, além de contribuir para a resiliência de várias maneiras.

Fallback significa "retroceder". No contexto de resiliência de aplicações, *fallback* refere-se a uma estratégia ou mecanismo alternativo que uma aplicação pode adotar quando ocorre uma falha ou quando um recurso esperado não está disponível. É uma técnica fundamental para garantir que a aplicação possa continuar a funcionar de maneira aceitável, mesmo diante de situações adversas.

Gabarito: Certo

Lista de Questões - Resiliência - CEBRASPE

1. (CEBRASPE/SERPRO/2023) Quanto à resiliência de aplicações, referente à capacidade de um aplicativo de fornecer seus serviços normalmente mesmo diante de situações adversas, julgue o item a seguir.

   Arquiteturas de site ativo × ativo são utilizadas para garantir a continuidade dos negócios e fornecer aos usuários uma experiência ininterrupta.

2. (CEBRASPE/MPO/2024) Julgue o item a seguir, relativo ao teste de estresse, a testes automatizados e à resiliência de aplicações.

   O uso de cache e a implementação de mecanismos de fallback são estratégias importantes para aumentar a resiliência de uma aplicação.

GABARITO

GABARITO

1- Certo

2- Certo

Servidores de Aplicação JEE (Java[1] Enterprise Edition)

Java é uma plataforma completa de desenvolvimento e execução, sendo composta de três elementos: a máquina virtual Java (JVM), um conjunto de APIs (*Application Programming Interfaces*) e a linguagem de programação Java (bastante utilizada na atualidade). Também é importante deixarmos claro que o "universo" Java é composto basicamente de três famílias:

- Java SE (*Standard Edition*): é a base da plataforma, incluindo o ambiente de execução e as bibliotecas Java;

- Java ME (*Micro Edition*): voltada para o desenvolvimento de aplicações para dispositivos móveis e embarcados;

- Java EE (*Enterprise Edition*): voltada para o desenvolvimento de aplicações corporativas e para internet.

Obviamente, o nosso foco será a especificação JEE (*Java Enterprise Edition*), além de APIs importantes no contexto dos servidores de aplicação, entre outros componentes. Bora lá!!!

Especificação JEE

Ao estudarmos a especificação JEE (chamado antigamente de J2EE), veremos alguns conceitos, tais como servlets, Beans, JSP, containers etc., os quais são essenciais para um melhor entendimento do assunto. Mas o que é a JEE? Trata-se de um conjunto de especificações destinadas a facilitar a criação de aplicações empresariais em Java. Ou seja, é uma "super especificação" que abrange outras especificações, tais como servlets, JSP, EJB, JDBC, JPA, JTA, JNDI, JMS, entre outras.

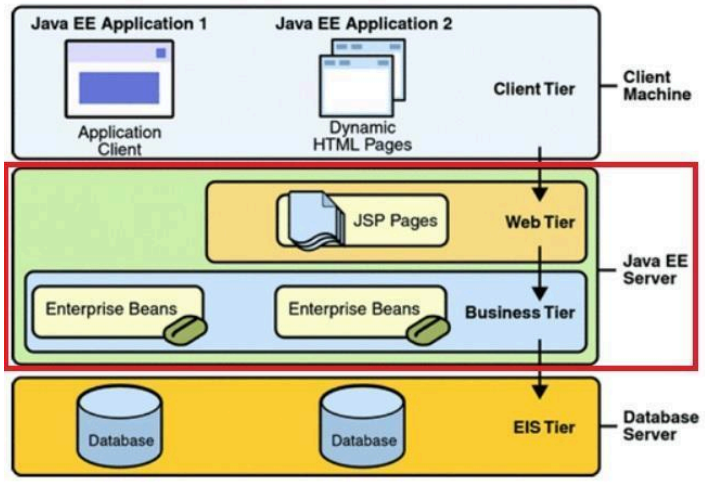
JEE define uma especificação modelo para criar aplicações, na qual atividades comuns (persistência de dados, validações, tratamento de requisições HTTP etc.) são especificadas e padronizadas, para os usuários lerem, implementarem e usarem. Também são definidos aspectos como alta disponibilidade, *clustering*, mensageria, entre outros. Além disso, ele também dispõe de APIs e *frameworks* (plataformas de desenvolvimento) para aplicações JEE escaláveis.

Se para cada aplicação criada houvesse a necessidade de implementar novamente aspectos como acesso a dados, segurança, entre outros, a vida dos administradores e dos programadores seria um caos! Você concorda? Por isso, essa é a proposta da especificação JEE: definir uma estrutura padronizada (necessária às aplicações corporativas) de forma reutilizável, o que acarreta na redução do retrabalho.

A especificação JEE disponibiliza bibliotecas que fornecem funcionalidades para implementar software Java distribuído, tolerante a falhas e multicamada, baseada amplamente em

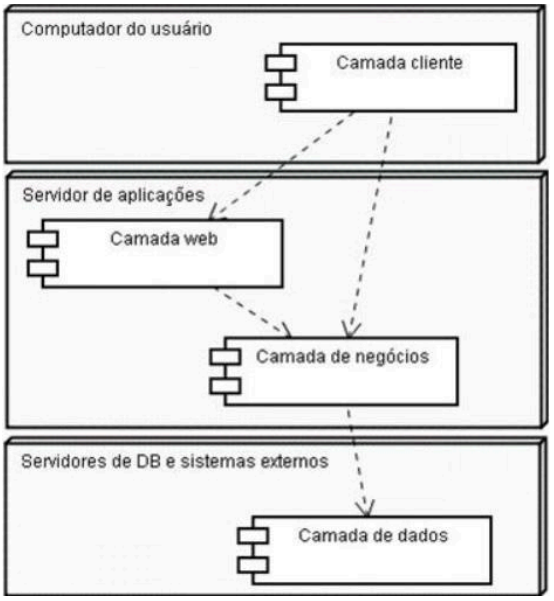
[1] Ao longo do tempo, a plataforma Java EE evoluiu e foi renomeada para Jakarta EE após ser transferida para a Fundação Eclipse. Portanto, os termos "Java EE Server" ou "Jakarta EE Server" podem ser usados para se referir a servidores que implementam as especificações mais recentes para o desenvolvimento de aplicativos corporativos Java. No texto desta aula ainda será mantido "Java", pois a maioria já se acostumou assim, inclusive as bancas com suas questões recentes!

componentes modulares, executados em um servidor de aplicações. A figura abaixo ilustra a arquitetura de um servidor de aplicações JEE (destaque em vermelho). Ainda nesta aula veremos as outras especificações (JSP, Beans etc.).



Camadas JEE

A J2EE define quatro camadas básicas no modelo de aplicação, ilustradas na figura abaixo e explicadas na sequência.



- Camada Cliente: Parte do software que roda no computador do usuário;
- Camada Web (ou de Apresentação): Parte do software que roda no servidor Web. Normalmente fica dentro do servidor de aplicações;
- Camada de negócios: Parte do software que roda no servidor de aplicações;
- Camada de dados: Banco de dados e sistemas externos.

APIs JEE

A especificação JEE define uma série de APIs, para facilitar a vida do programador. Vamos focar nas principais:

- JDBC (*Java Data Base Connectivity*): conjunto de interfaces escritas em Java que faz o envio de instruções SQL para banco de dados. Possibilita o uso de bancos de dados instalados, sendo que para cada banco de dados há um driver JDBC;

- JPA (*Java Persistence API*): *framework* para persistir objetos Java simples (POJOS - *Plain Old Java Objects*). Entre as principais características da JPA temos as consultas em objetos da *Java Persistence Query Language* (JPQL);

- JTA (*Java Transaction API*): especifica interfaces locais que são utilizadas entre um gerenciador de transação[2] e as partes que estão envolvidas em uma transação, como a aplicação, o gerenciador de recursos e o servidor de aplicação;

- JNDI (*Java Naming and Directory Interface*): API para acesso a serviços de nomes e diretórios, permitindo que aplicações cliente descubram e obtenham dados ou objetos através de um nome (semelhante ao serviço DNS, na Internet). A JNDI é utilizada em aplicações Java que acessam recursos externos, como base de dados, filas, entre outros;

- JMS (*Java Message Service*): uma especificação e um sistema de mensageria para servidores de aplicação JEE. Seus principais componentes são *JMS Provider*, *JMS Clients* e *Administered Objects*;

- EJB (Enterprise JavaBeans): define um modelo de programação para a construção de componentes de negócios em Java, facilitando o desenvolvimento de aplicações corporativas distribuídas e escaláveis.

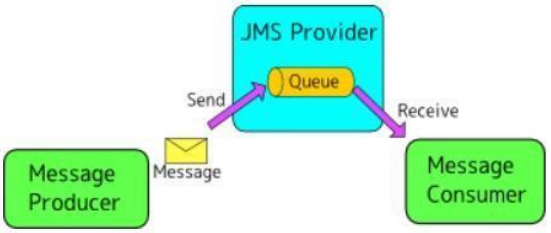
Há situações em que as aplicações precisam se comunicar de forma assíncrona com outras aplicações ou com um componente dentro do mesmo aplicativo. Por exemplo, algumas operações podem ser registradas e pode ser preferível que o log seja escrito de forma assíncrona, em vez de aguardar que a transação seja concluída. Os sistemas de mensageria surgiram para resolver esse tipo de questão.

Mensagens são uma forma de comunicação entre softwares distribuídos de forma fracamente acoplada (componentes ou módulos têm interações mínimas entre si, sendo que as mudanças em um componente têm impacto limitado nos outros). Em comunicação por mensagens, o remetente envia uma mensagem ao destinatário, porém ambos não precisam estar disponíveis ao mesmo tempo ("offline", assim como funciona o envio de mensagens por e-mail). O remetente não precisa saber nada sobre o destinatário, precisa saber apenas qual o formato da mensagem e qual o destino.

[2] Transação: sequência de operações que são tratadas como um bloco único e indivisível e segue basicamente quatro princípios: atomicidade, consistência, isolamento e durabilidade.

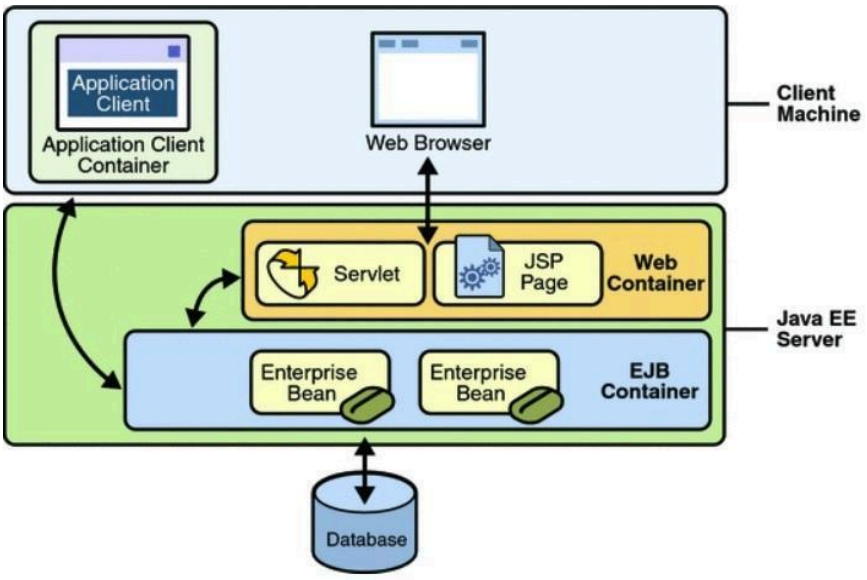
A lógica da aplicação permite que seja enviada uma mensagem e ela continue operando sem receber uma resposta imediatamente. Alguns conceitos recorrentes quando tratamos de serviços de mensageria são:

- *Producer/Publisher JMS*: cria e envia mensagens;
- *Consumer/Subscriber JMS*: recebe mensagens;
- *Destination*: objeto que o cliente usa para especificar o alvo das mensagens que produz e a origem de mensagens que consome;
- *JMS Queue*: fila ou área contendo mensagens que aguardam para serem lidas. As mensagens não são necessariamente lidas na ordem que a fila propõe;
- *JMS Topic*: mecanismo de publicação de mensagens para múltiplos leitores;
- *Publish/Subscribe*: uma mensagem que é enviada a um conjunto de destinatários;
- *Point to point (Queue)*: uma mensagem que é enviada a um único destinatário.



Containers Web e EJB

Vamos analisar agora os containers Web e EJB, que ficam dentro do servidor JEE, como podemos ver na figura abaixo.

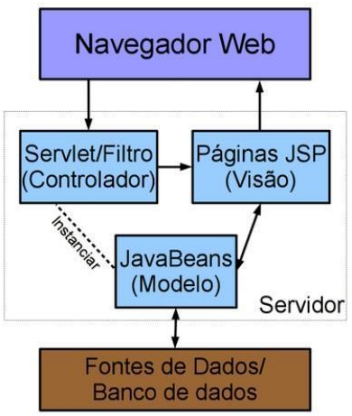


Web Container

Servlets: são utilizados para o desenvolvimento de aplicações Web com conteúdo dinâmico. Contém uma API que abstrai e disponibiliza os recursos do servidor Web (ex.: Apache, IIS, Nginx etc.) de maneira simplificada para o programador. Resumindo, servlet é basicamente uma classe na linguagem de programação Java que dinamicamente processa requisições e respostas, proporcionando novos recursos aos servidores.

As aplicações baseadas em servlet geram conteúdo dinâmico (geralmente HTML) e interagem com os clientes, utilizando o modelo request/response, normalmente através do protocolo HTTP. Um Servlet necessita de um container Web para ser executado.

JSP (Java Server Page): é uma extensão da tecnologia Servlet. Trata-se de um documento convertido pelo container JSP em um servlet. JSP permite que os programadores Web possam desenvolver conteúdos dinâmicos com a reutilização de componentes predefinidos com interação de componentes que utilizam script do lado do servidor. JSP é uma especialização do servlet que permite que conteúdo dinâmico seja facilmente desenvolvido. Ou seja, enquanto o foco do EJB é em regras de negócio, o foco da JSP é na apresentação:



Fazendo um comparativo, JSP é similar às tecnologias Active Server Pages (ASP) da Microsoft e ao PHP, ou seja, é utilizada no desenvolvimento de aplicações dinâmicas para Web. Uma página criada com a tecnologia JSP, após instalada em um servidor de aplicação compatível com a tecnologia JEE, é transformada em um servlet.

JSF (JavaServer Faces): *framework* Java para o desenvolvimento de interfaces de usuário baseadas em componentes para aplicativos web. Em um contexto JSF, um "Managed Bean" é uma classe Java associada a uma página JSF e gerenciada pelo *framework*. Os Managed Beans

são usados para armazenar e gerenciar dados que são exibidos ou processados em uma página JSF.

EJB (Enterprise Java Beans)

O EJB (*Enterprise JavaBeans*) é um dos principais componentes da plataforma JEE, pois nele é implementada a lógica de negócio. É um componente do tipo servidor que executa no container do servidor de aplicação. Os EJBs permitem que o desenvolvedor se concentre nas necessidades do negócio do cliente, enquanto questões de infraestrutura, segurança, disponibilidade e escalabilidade ficam sob responsabilidade do servidor de aplicações. EJBs são utilizados no desenvolvimento de componentes de software, e normalmente são encapsulados no container EJB.

Os principais objetivos da tecnologia EJB são fornecer um rápido e simplificado desenvolvimento de aplicações Java baseado em componentes distribuídos, transacionais, seguros e portáveis.

Message-Driven Beans (MDBs): componentes usados para processar mensagens assíncronas, particularmente úteis em cenários de integração de sistemas onde a comunicação entre diferentes partes do sistema é baseada em mensagens. Fazem parte do modelo de programação de Enterprise JavaBeans (EJB) e são usados em conjunto com sistemas de mensagens, como o Java Message Service (JMS).

Stateful x Stateless Session Beans

- *Stateful Session Beans* (Beans de Sessão com Estado) são componentes que mantêm um estado conversacional com um cliente ao longo de várias chamadas de método. Isso quer dizer que esses *beans* retêm informações específicas do cliente entre chamadas de método e sessões. Para manter o estado do cliente, é necessário considerar os impactos de desempenho, especialmente se houver muitas instâncias sendo mantidas simultaneamente. Essa abordagem é útil em cenários onde é necessário manter um estado específico do cliente no servidor;

- *Stateless Session Beans* (Beans de Sessão Sem Estado) são componentes que oferecem uma lógica de negócios sem a necessidade de manter um estado específico entre as chamadas de método para um cliente. Ou seja, não mantêm informações sobre o estado do cliente entre chamadas de método. Cada chamada de método para um *Stateless Session Bean* é independente (não depende de chamadas anteriores). Por não haver estado, os *Stateless Session Beans* geralmente possuem melhor desempenho do que os *Stateful Session Beans*. São mais adequados para cenários em que não é necessário manter um estado entre chamadas.

Container JEE

Um container é um objeto que contém outros objetos, no contexto da especificação JEE. Podemos pensar em um "grande objeto" que possui outros objetos. Associando ao nome "container", fica mais fácil para lembrar. Esses objetos podem ser incluídos ou removidos dinamicamente, em tempo de execução, diferentemente do que ocorre com uma composição onde esse relacionamento é fixado em tempo de compilação. Um container é uma interface entre um componente e uma funcionalidade de baixo nível de uma plataforma.

Os cinco containers JEE são:

- JEE *Server:* ambiente de execução que suporta várias especificações, facilitando o desenvolvimento, implantação e execução de aplicativos corporativos. Algumas das especificações que um JEE Server pode implementar incluem: Servlet API, JSP, EJB, JPA, JMS, JTA, JCA.

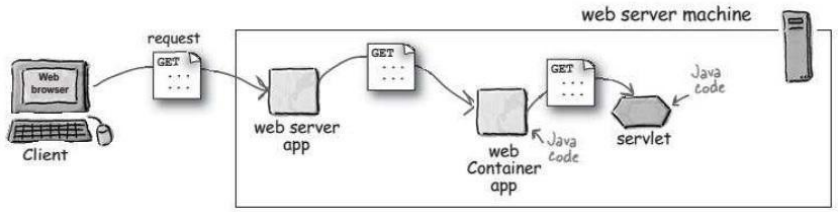
- *Container* EJB: gerencia a execução de Enterprise Beans (os modelos de negócio);

- Container Web: gerencia a execução de páginas JSP (aquelas dinâmicas) e componentes servlet para aplicações Java EE;

- *Application client* container: gerencia a execução dos componentes da aplicação cliente. Aplicações clientes e seus containers rodam no cliente (aplicações Java standalone);

- Servlet container: gerencia a execução de servlets. O container gerencia o ciclo de vida, dá suporte ao *multithread*, segurança, e suporte para páginas JSP, no caso dos containers web.

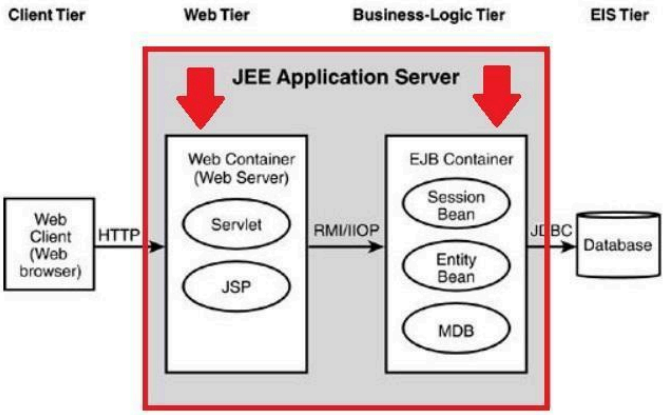
Quando uma aplicação Web recebe uma requisição para um servlet, o servidor Web (o Apache, por exemplo) a encaminha não diretamente ao servlet, mas para o container do servlet:



Em JEE, um container pode conter e gerenciar componentes como Servlets (servlet container - para aplicações Web) ou EJBs (EJB container - para componentes de negócio).

Servidores de Aplicação JEE

Servidores de aplicação JEE completam a especificação JEE, e disponibilizam uma plataforma de middleware baseada em padrões abertos, em conformidade com a especificação JEE. Um servidor de aplicação JEE é também chamado servidor *full compliance*. Eles implementam o web container e o EJB container:



Assim, um servidor de aplicações totalmente compatível com a especificação JEE (*full compliance*) engloba tanto um container web, como um container EJB. Por exemplo, o Tomcat não é um servidor de aplicações JEE full, pois implementa apenas Servlets e JSP.

E qual é o conceito de servidor de aplicações? Servidores de aplicação JEE são sistemas que fornecem a infraestrutura de serviços para a execução de aplicações distribuídas. Resumindo: servidor de aplicação é um software que fornece um ambiente para a execução das aplicações.

Um servidor de aplicação disponibiliza um ambiente para a instalação e a execução de aplicações. Os servidores de aplicação também são conhecidos como middleware (faz o "meio de campo"). O foco do servidor de aplicações é a resolução de problemas relacionados ao negócio, e não de questões de infraestrutura da aplicação. O servidor de aplicações cuida de algumas questões comuns a todas as aplicações, tais como segurança, garantia de disponibilidade, balanceamento de carga, entre outras.

Servidores de aplicação representam a borda entre a programação e a infraestrutura, por isso o seu estudo é bem abrangente, e incluem os conceitos relacionados a JEE. Seu principal papel é servir a aplicação com serviços de infraestrutura (aquela camada mais abaixo, a base para um bom funcionamento).

Com toda essa base pronta, fica mais fácil o compartilhamento de componentes e aplicações, o desenvolvimento, a manutenção e o gerenciamento de sistemas complexos. Algumas das principais facilidades providas por um servidor de aplicações são:

- Gerenciamento: servidor auxilia a decidir se a máquina deve ser desligada ou mantida em execução em situações como, por exemplo, a realização de atualizações de software enquanto o site é executado;

- Gerenciamento de transações: lida com problemas como o acesso simultâneo à mesma linha do banco de dados ou quando o banco de dados "cai";

- Segurança: gerência de permissões de usuários para executar operações, registro de log para auditoria, entre outros;

- Tolerância a falhas: lida com situações em que um servidor "cai" ou um serviço falha, por exemplo, para que o serviço não pare de funcionar;

- Balanceamento de carga: direciona os usuários ao servidor com menos carga. Se um servidor estiver sobrecarregado, outro servidor poderá ser escolhido;

- Failover: se um servidor deixar de funcionar, os clientes poderão ser redirecionados para outros servidores sem a interrupção de serviço. Se isso for possível, é importante que seja o mais rápido possível, e essa definição de tempo aceitável deveria ocorrer de acordo com os requisitos de negócio (1 minuto? 1 hora? etc.);

- Mensagens: alguns tipos de solicitações devem ser baseadas em mensagem, um sistema em que os clientes e servidores estão muito fracamente acoplados. Portanto, é necessária uma infraestrutura para acomodar um sistema de mensagens;

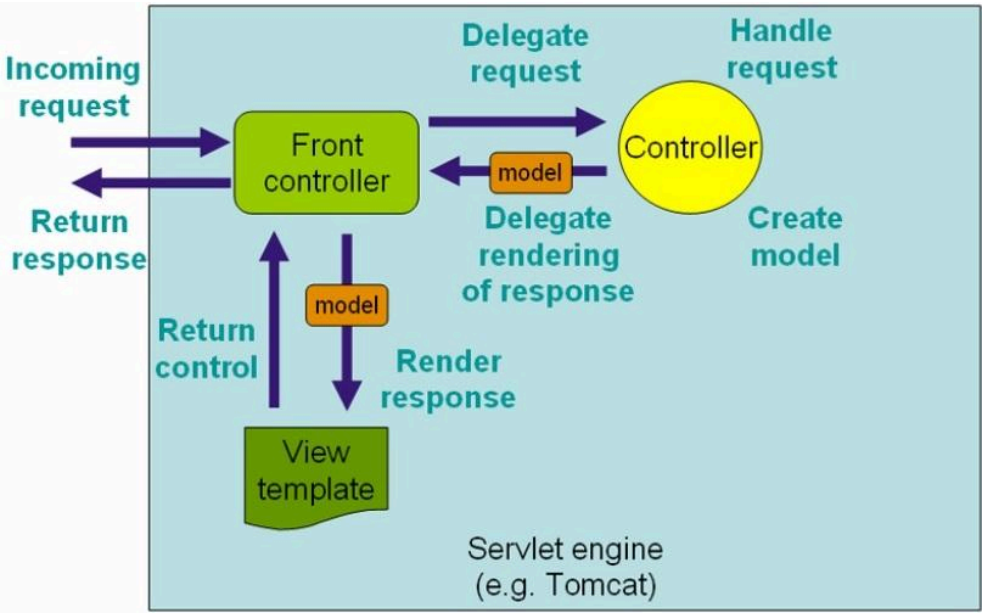
- *Pool* de recursos: quando um cliente não estiver utilizando um servidor, esses recursos preciosos do servidor poderão ser retornados para um pool, a fim de serem reutilizados quando outros clientes se conectarem. Exemplos de recursos: soquetes (como conexões de banco de dados), objetos que "vivem" dentro do servidor etc.

Para finalizar esse assunto, é importante saber que existem diversos exemplos de servidores de aplicação que implementam a especificação JEE, tais como: JBoss, Weblogic, Glassfish, OAS, Geronimo, WebSphere, Netweaver, entre outros.

Outros Conceitos

Front Controller

Em arquiteturas de software Java, o padrão Front Controller é uma abordagem que centraliza o controle da lógica de manipulação de requisições em uma única entrada (o nome sugere isso, um controlador "frontal"). A finalidade é ter um ponto centralizado para o processamento de solicitações em uma aplicação Web, permitindo maior modularidade e facilitando a manutenção. Abaixo uma ideia de como funciona.



A implementação de um Front Controller geralmente envolve os elementos abaixo.

- Front Controller: Uma classe ou componente que recebe todas as requisições de entrada e coordena o fluxo de controle para a aplicação. Pode ser um servlet;

- Roteamento: O Front Controller geralmente inclui um mecanismo de roteamento que determina qual controlador específico (*controller*) deve ser invocado com base na requisição recebida ("delegate request" na figura). Pode ser realizado por meio de mapeamentos de URL, anotações, configurações de arquivo etc.;

- Controladores: Componentes responsáveis por lidar com solicitações específicas. Cada tipo de solicitação pode ter seu próprio controlador. Podem ser classes Java, métodos, ou outros componentes que processam a lógica de negócios associada à requisição;

- View: Após o processamento da lógica de negócios, o Front Controller geralmente encaminha a resposta para uma visualização (view) apropriada para a renderização da resposta final. Pode ser uma página JSP, um modelo de página HTML, um componente de renderização JSF etc.;

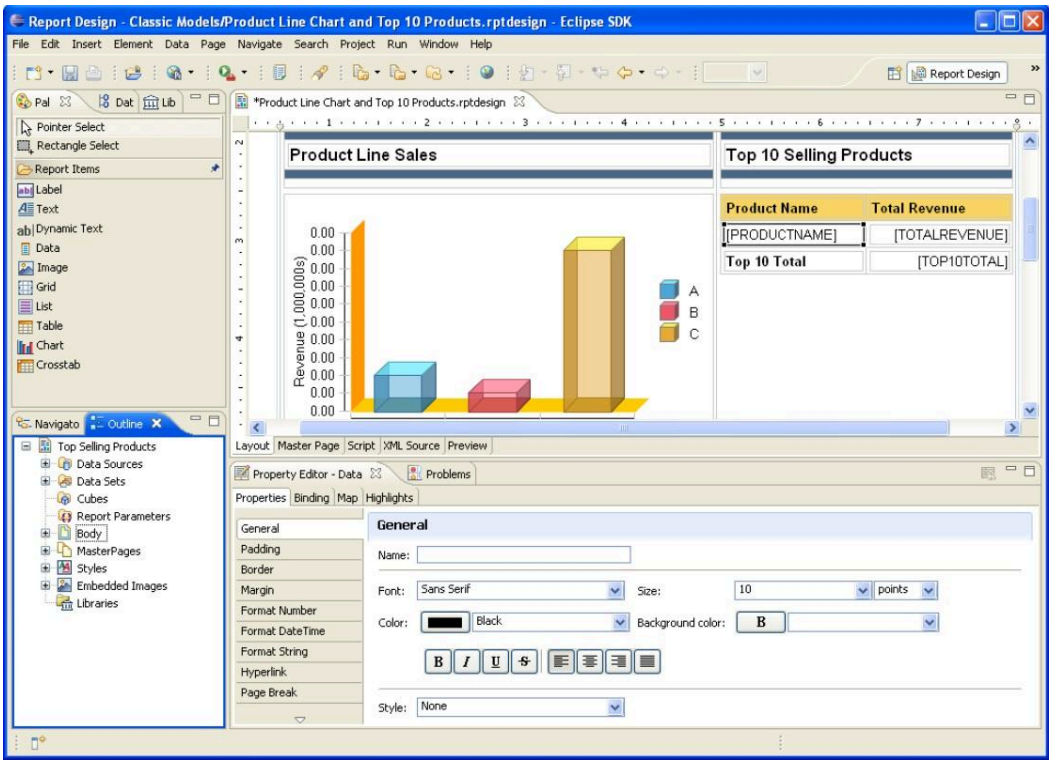
- Configuração: Pode haver uma configuração centralizada para mapear URLs para controladores, definir interceptadores, e especificar outras configurações relacionadas ao fluxo de controle.

Data Access Object (DAO): Padrão de design comumente utilizado para separar a lógica de acesso a dados do restante da aplicação. O DAO é responsável por fornecer uma interface abstrata para a comunicação com uma fonte de dados (geralmente se utiliza um banco de dados) e encapsular a complexidade da manipulação de dados.

EclipseLink: Provedor de persistência de código aberto e um framework de mapeamento objeto-relacional (ORM) para Java. É uma implementação da especificação JPA (Java Persistence API) e fornece funcionalidades avançadas para mapeamento objeto-relacional, consultas JPQL (Java Persistence Query Language), cache de segundo nível, entre outros itens relacionados à persistência de dados em aplicativos Java.

Para utilizar o EclipseLink em um projeto Java, normalmente é necessário incluir suas bibliotecas no caminho das classes (*classpath*) do projeto e configurar um arquivo de persistência (persistence.xml) para especificar as configurações de persistência, como a unidade de persistência, provedor JPA (EclipseLink, neste caso), e detalhes de conexão com o banco de dados.

BIRT (Business Intelligence and Reporting Tools): Plataforma de código aberto para desenvolvimento de relatórios e inteligência empresarial. Mantido pela Eclipse Foundation, oferece ferramentas poderosas para criar, visualizar e gerenciar relatórios de negócios em diversos formatos. É amplamente utilizado em várias aplicações e ambientes para fornecer recursos de relatórios avançados.



Referência Bibliográfica

Java EE at a Glance. Disponível em <https://www.oracle.com/br/java/technologies/java-ee-glance.html>.

QUESTÕES COMENTADAS - SERVIDORES DE APLICAÇÃO JEE (JAVA ENTERPRISE EDITION) - MULTIBANCAS

1. (FCC/TRT1 - 2011) J2EE é uma plataforma de programação para servidores na linguagem de programação Java, que integra uma série de especificações e containers, cada uma com funcionalidades distintas.

   Nesse contexto, é correto afirmar que são integrantes do J2EE:

A) Servlets, Jcompany e JSP.

B) JDBC, JSP, EJBs.

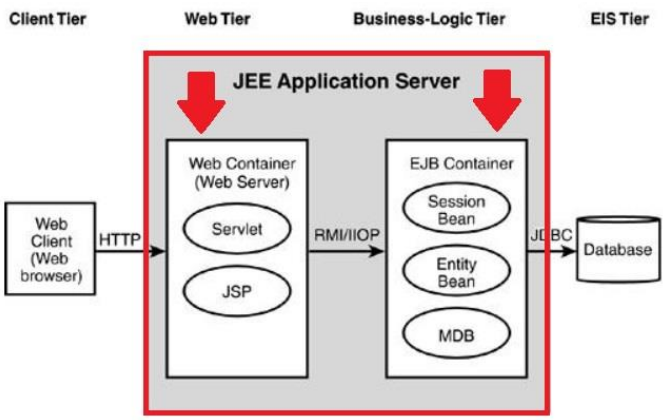
C) EJBs, Servlets e JBoss.

D) JDBC, Hibernate e JPA.

E) JSP, JSF e Eclipse.

Comentários:

Questão antiga, ainda chamava de J2EE, mas tudo bem. Servlets, JSPs e EJBs integram a especificação JEE. O JBoss é um exemplo de implementação de servidor de aplicação que implementa totalmente a especificação, cuidado! Vamos relembrar a figura abaixo. Podemos ver os servlets e JSP dentro do Web container e os EJBs ficam dentro de um EJB container.



Portanto, a alternativa B está correta e é o gabarito da questão.

Gabarito: Letra B

2. (CESGRANRIO/FINEP - 2011) No contexto da plataforma Java EE (Enterprise Edition), como é chamada a entidade de software que fornece uma interface entre um componente e as funcionalidades de baixo nível específicas da plataforma que suporta tal componente?

A) Servlet

B) Class

C) JSP

D) Service

E) Container

Comentários:

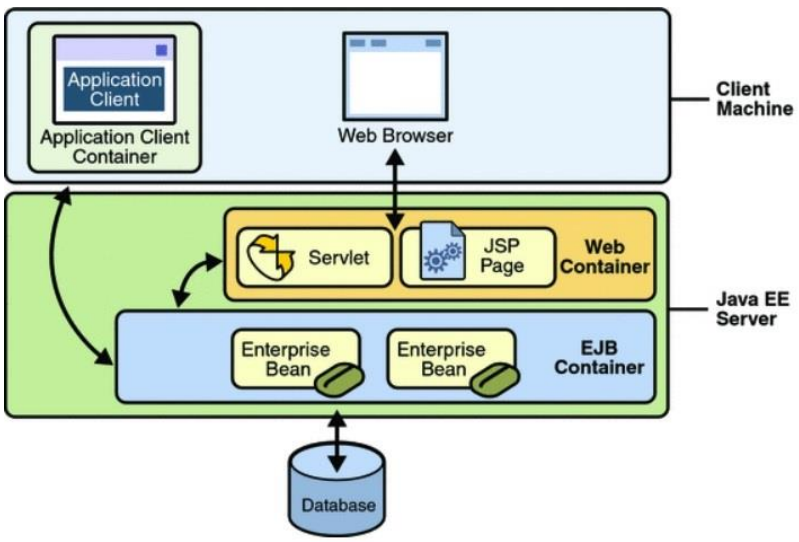
Container é um recipiente de objetos Java, ou, de uma maneira mais formal, entidade de software que fornece uma interface entre um componente e as funcionalidades de baixo nível específicas da plataforma Java. Portanto, a alternativa E está correta e é o gabarito da questão.

Gabarito: Letra E

3. (FCC/TJ-PE - 2012) Sobre JEE e tecnologias relacionadas é correto afirmar que um servidor de aplicações Java EE possui um único contêiner conhecido como contêiner EJB. Além disso, Serlvets e JSP rodam no contêiner EJB do servidor de aplicação JEE.

Comentários:

Um servidor de aplicações JEE não possui um único contêiner. Vamos ver outra figura:



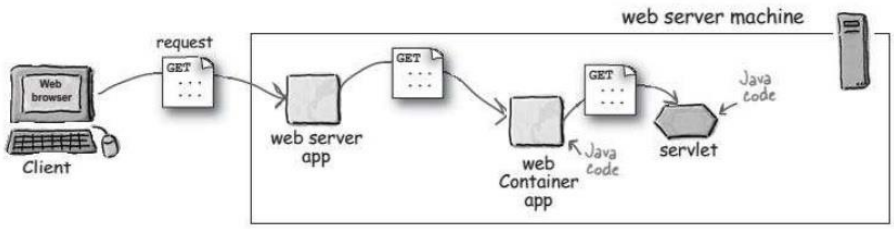
Veja que existem o contêiner Web (contendo Servlets e páginas JSP) e o contêiner EJB (contendo as lógicas de negócio). Portanto, a questão está errada.

Gabarito: Errada

4. (FCC/TRE-CE - 2012) No contexto do ciclo de vida de um servlet, quando o servidor recebe uma requisição, ela é repassada para o container que, por sua vez, carrega a classe na memória e cria uma instância da classe do servlet.

Comentários:

Quando uma aplicação Web recebe uma requisição para um servlet, o servidor Web (o Apache, por exemplo) a encaminha não diretamente ao servlet, mas para o container do servlet:



No ciclo de vida dos servlets, quando o servidor recebe uma requisição, ele não a processa diretamente, nem repassa a requisição para o servlet. Na verdade, a requisição é repassada para o container, que as gerencia. Um dos ganhos obtidos com o container é este, ter um objeto que passa a gerenciar o ciclo de vida de um componente. Portanto, a questão está correta.

Gabarito: Correta

5. (CESGRANRIO/LIQUIGAS - 2012) Devido à popularização da plataforma Java, o termo "servidor de aplicação" é frequentemente usado como um sinônimo de "servidor de aplicação J2EE". Nesse contexto, os servidores a seguir implementam a especificação J2EE, EXCETO

A) WebSphere Application Server

B) Oracle WebLogic Server

C) Glassfish

D) JBoss AS

E) IIS

Comentários:

Questão mais antiga, ainda utilizava o nome J2EE, mas tudo bem. O JBoss é um servidor de aplicação ou servidor de aplicação JEE, ou seja implementa totalmente a especificação JEE. O JBoss, WebSphere, WebLogic e Glassfish também são servidores que implementam a especificação JEE. E o IIS (Information Internet Services)? Bom, esse é um servidor Web da Microsoft! Portanto, a alternativa E está correta e é o gabarito da questão.

Gabarito: Letra E

6. (IBFC/TRE-AM - 2014) A plataforma JEE (Java Platform, Enterprise Edition) contém uma série de especificações e containers, cada uma com funcionalidades distintas, o Container utilizado no acesso ao banco de dados é conhecido como:

A) JSP

B) JABD

C) JMS

D) JDBC

Comentários:

JEE é uma "super especificação" que engloba outras especificações. Entre elas, vimos que o JEE especifica várias APIs, para persistência (JPA), transações (JTA), mensageria (JMS), entre outras. A API que define conexão a bancos de dados é a JDBC. Com esse monte de "letrinhas", note o DB no meio da sigla, aí fica mais fácil associar com "Data Base" = Banco de Dados. Portanto, a alternativa D está correta e é o gabarito da questão.

Gabarito: Letra D

7. (FUNCA/MDA - 2014) Um servidor de aplicação JAVA EE pode ser considerado mais completo que um Web Container, pois além de fornecer um Web Container, esse servidor também fornece um:

A) Servlet Container.

B) JAAS Container.

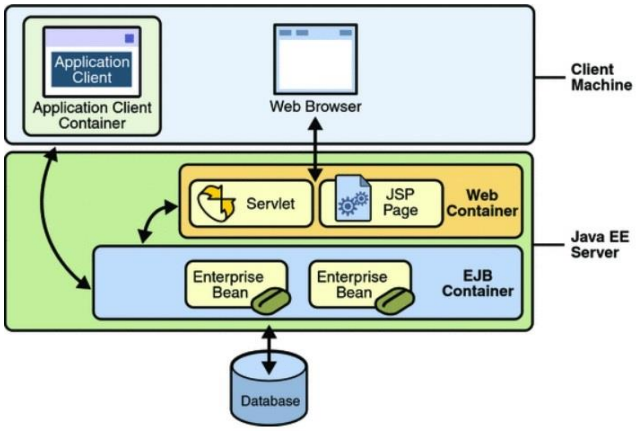
C) JSP Container.

D) EJB Container.

E) JDBC Container.

Comentários:

Questão fácil, depois de vermos algumas...não é? Resumindo: Servidor de aplicação JEE = Web container (Servlets, JSP) + EJB container. Para reforçar, aquela figura:



Portanto, a alternativa D está correta e é o gabarito da questão.

Gabarito: Letra D

8. (FCC/TCM-GO - 2015) Um Analista de Controle Externo da área de TI do TCM/GO está trabalhando em uma aplicação web utilizando a plataforma Java EE. Ciente que os componentes desta aplicação, para serem processados no servidor de aplicações, terão que ser implantados (deployed) em contêineres apropriados, ele esclareceu à equipe de desenvolvimento que servlets, JavaServer Pages, páginas HTML e Java Beans serão implantados e processados no contêiner A. Além disso, alguns componentes serão implantados e processados no contêiner B, destinado a prover a infraestrutura necessária para a execução de componentes de negócio distribuídos que estendem as funcionalidades de um servidor, permitindo encapsular lógica de negócio e dados específicos de uma aplicação. Os contêineres A e B são conhecidos, respectivamente, como

A) local container e remote container.

B) web container e EJB container.

C) glassfish container e tomcat container.

D) EJB container e web container.

E) server container e client container.

Comentários:

Servlets, Java Server Pages (JSP), páginas HTML e Java Beans são implantados e processados no Web container, responsável por gerir o ciclo de vida. O EJB container permite encapsular lógica de negócio e dados específicos de uma aplicação. Portanto, a alternativa B está correta e é o gabarito da questão.

Gabarito: Letra B

9. (CESPE/ABIN - 2018) Uma das finalidades de um servidor de aplicação web é fornecer facilidades para que o desenvolvedor seja capaz de lidar com a heterogeneidade das especificações de hardware dos servidores.

Comentários:

O servidor de aplicação abstrai modelos e dados oriundos da camada de dados para atender requisições Web. Então, sua finalidade é fornecer facilidades aos desenvolvedores no acesso aos dados de negócio. "Quem" tem que lidar com o hardware é o sistema operacional! Portanto, a questão está errada.

Gabarito: Errada

10. (CESPE/ABIN - 2018) Em servidores de aplicação, cache diminui significativamente a carga em banco de dados, especialmente em aplicações que somente realizam leitura no banco, assim, cache em memória é melhor que cache em disco, que é melhor que um banco de dados remoto.

Comentários:

O acesso aos dados em aplicações que realizam muitas consultas (leituras) pode constituir um gargalo. Para reduzir essa possibilidade de gargalo é possível adotar uma estratégia de cache em memória, dos dados mais frequentemente ou recentemente acessados. A cache em memória possui tempos de acesso menores que o acesso em cache em disco ou em bancos de dados remotos. Portanto, a questão está correta.

Gabarito: Correta

11. (COSEAC/UFF - 2019) O J2EE é um exemplo de aplicação potencialmente utilizada em:

A) VPN.

B) web services.

C) PKI.

D) RSS.

E) SQL.

Comentários:

Mesmo sendo uma questão recente, ainda com o nome antigo! Vimos que o JEE é um servidor de aplicações, sendo um middleware (software que fornece serviços para softwares aplicativos além daqueles disponíveis pelo sistema operacional), ou seja, dando uma base para os aplicativos. Quando se fala em middleware, um padrão bastante utilizado são os Web Services! E para completar, as outras opções são siglas ligadas à segurança (VPN - Virtual Private Network, PKI - Public Key Infraestructure), à sites (RSS - Rich Site Summary) ou acesso a banco de dados (SQL – Structured Query Language). Portanto, a **alternativa B** está correta e é o gabarito da questão.

Gabarito: Letra B

Lista de Questões - Servidores de Aplicação JEE (Java Enterprise Edition) - Multibancas

1. (FCC/TRT1 - 2011) J2EE é uma plataforma de programação para servidores na linguagem de programação Java, que integra uma série de especificações e containers, cada uma com funcionalidades distintas.

   Nesse contexto, é correto afirmar que são integrantes do J2EE:

A) Servlets, Jcompany e JSP.

B) JDBC, JSP, EJBs.

C) EJBs, Servlets e JBoss.

D) JDBC, Hibernate e JPA.

E) JSP, JSF e Eclipse.

2. (CESGRANRIO/FINEP - 2011) No contexto da plataforma Java EE (Enterprise Edition), como é chamada a entidade de software que fornece uma interface entre um componente e as funcionalidades de baixo nível específicas da plataforma que suporta tal componente?

A) Servlet

B) Class

C) JSP

D) Service

E) Container

3. (FCC/TJ-PE - 2012) Sobre JEE e tecnologias relacionadas é correto afirmar que um servidor de aplicações Java EE possui um único contêiner conhecido como contêiner EJB. Além disso, Serlvets e JSP rodam no contêiner EJB do servidor de aplicação JEE.
4. (FCC/TRE-CE - 2012) No contexto do ciclo de vida de um servlet, quando o servidor recebe uma requisição, ela é repassada para o container que, por sua vez, carrega a classe na memória e cria uma instância da classe do servlet.

5. (CESGRANRIO/LIQUIGAS - 2012) Devido à popularização da plataforma Java, o termo "servidor de aplicação" é frequentemente usado como um sinônimo de "servidor de aplicação J2EE". Nesse contexto, os servidores a seguir implementam a especificação J2EE, EXCETO

A) WebSphere Application Server

B) Oracle WebLogic Server

C) Glassfish

D) JBoss AS

E) IIS

6. (IBFC/TRE-AM - 2014) A plataforma JEE (Java Platform, Enterprise Edition) contém uma série de especificações e containers, cada uma com funcionalidades distintas, o Container utilizado no acesso ao banco de dados é conhecido como:

A) JSP

B) JABD

C) JMS

D) JDBC

7. (FUNCA/MDA - 2014) Um servidor de aplicação JAVA EE pode ser considerado mais completo que um Web Container, pois além de fornecer um Web Container, esse servidor também fornece um:

A) Servlet Container.

B) JAAS Container.

C) JSP Container.

D) EJB Container.

E) JDBC Container.

8. (FCC/TCM-GO - 2015) Um Analista de Controle Externo da área de TI do TCM/GO está trabalhando em uma aplicação web utilizando a plataforma Java EE. Ciente que os componentes desta aplicação, para serem processados no servidor de aplicações, terão que

ser implantados (deployed) em contêineres apropriados, ele esclareceu à equipe de desenvolvimento que servlets, JavaServer Pages, páginas HTML e Java Beans serão implantados e processados no contêiner A. Além disso, alguns componentes serão implantados e processados no contêiner B, destinado a prover a infraestrutura necessária para a execução de componentes de negócio distribuídos que estendem as funcionalidades de um servidor, permitindo encapsular lógica de negócio e dados específicos de uma aplicação. Os contêineres A e B são conhecidos, respectivamente, como

A) local container e remote container.

B) web container e EJB container.

C) glassfish container e tomcat container.

D) EJB container e web container.

E) server container e client container.

9. (CESPE/ABIN - 2018) Uma das finalidades de um servidor de aplicação web é fornecer facilidades para que o desenvolvedor seja capaz de lidar com a heterogeneidade das especificações de hardware dos servidores.

10. (CESPE/ABIN - 2018) Em servidores de aplicação, cache diminui significativamente a carga em banco de dados, especialmente em aplicações que somente realizam leitura no banco, assim, cache em memória é melhor que cache em disco, que é melhor que um banco de dados remoto.

11. (COSEAC/UFF - 2019) O J2EE é um exemplo de aplicação potencialmente utilizada em:

A) VPN.

B) web services.

C) PKI.

D) RSS.

E) SQL.

GABARITO

GABARITO

1- B
2- E
3- Errada
4- Correta
5- E
6- D
7- D
8- B
9- Errada
10- Correta
11- B

Servidor de Aplicação JBoss/Wildfly

JBoss é um servidor de aplicação de código fonte aberto baseado na plataforma JEE, é implementado completamente na linguagem de programação Java, podendo ser utilizado em qualquer sistema operacional que suporte essa linguagem.

O JBoss Application Server (AS) é a versão *open source*, desenvolvida em comunidades JBoss/RedHat, traz inovações em um ritmo mais rápido com foco em novas funcionalidades, mas seu principal diferencial é que ele não possui suporte oficial.

O JBoss Enterprise Application Platform (EAP) é a versão paga, normalmente evolui a partir das inovações das versões estáveis do JBoss AS, é integrada com recursos como o JBoss Developer Studio e o com o JBoss Operations Network, após um processo de testes (desempenho, escalabilidade etc.), além de possuir suporte oficial.

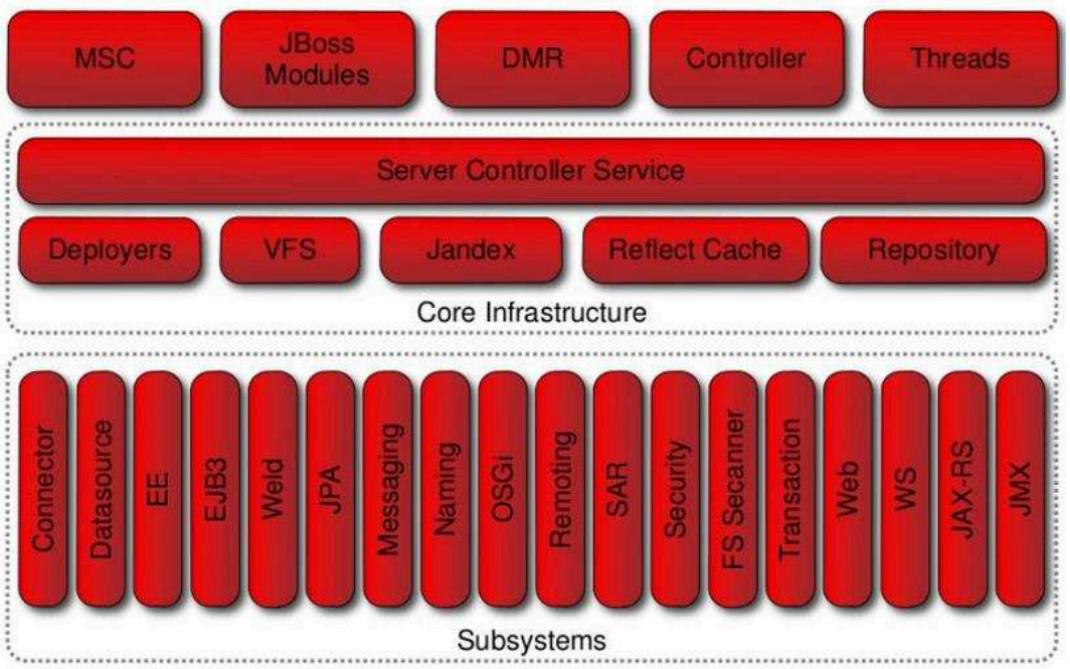
O JBoss é uma plataforma de middleware[1] baseada em padrões abertos, e que mantém conformidade com a especificação Java EE. É servidor de aplicação que provê recursos de alta disponibilidade, *clustering*, mensageria, *cache* distribuído, entre outros. Além disso, ele também inclui APIs e *frameworks* de desenvolvimento para aplicações Java EE escaláveis.

A partir da versão 8 o JBoss passou a se chamar Wildfly, além de ter várias melhorias e mudanças como a troca do *container* que era o JBossWeb para o Undertow e a utilização de módulos. O objetivo principal dos módulos é fornecer um mecanismo de isolamento e organização para os componentes do servidor e das aplicações implantadas. O JBoss Modules permite que os módulos sejam carregados e descarregados dinamicamente, o que torna a utilização de recursos otimizada.

Arquitetura e Modos

Na figura a seguir podemos ver a arquitetura do JBoss/Wildfly, cabendo um destaque aos subsistemas, onde encontramos algumas APIs JEE, tais como EJB, JPA, JMX, Mensageria, entre outros.

[1] Software que se encontra entre o sistema operacional e os aplicativos nele executados, funcionando de forma essencial como uma camada oculta de tradução. Permite a comunicação e o gerenciamento de dados para aplicativos distribuídos.



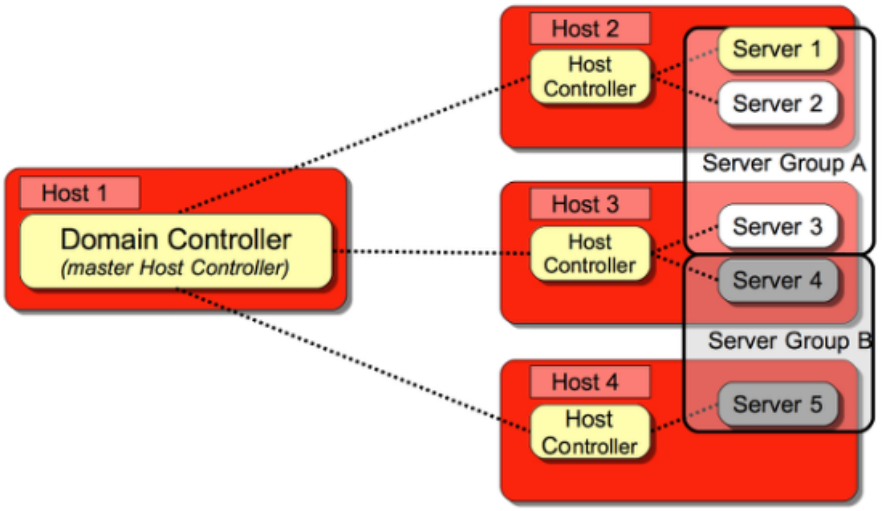
Com essa figura fica claro que o JBoss é um servidor de aplicação que implementa diversas características da especificação JEE. Um conceito essencial nas últimas versões é o de domínio. Em um servidor JBoss, um domínio fornece gerenciamento centralizado de várias instâncias de servidor e hosts físicos, enquanto um servidor standalone permite uma única instância do servidor. Configurações, implantações, socket, módulos, extensões e propriedades do sistema podem ser gerenciadas para grupos de servidores. Vamos ver mais detalhes desses dois modos a seguir.

Modo Standalone: é o modo tradicional das versões anteriores. Basicamente implica em ter uma instalação diferente (ou um diretório standalone diferente) para cada instância de Wildfly. Ou seja, para cada Wildfly em execução no seu ambiente é necessário alterar seus próprios arquivos de configuração, suas próprias opções de execução para JVM etc. A inicialização nesse modo é realizada ao iniciar o script JBOSS_HOME/standalone.sh (no Linux) ou JBOSS_HOME/standalone.bat (no Windows).

Modo Domain: é o modo que foi introduzido no JBoss AS 7, permitindo gerenciar um conjunto de instâncias Wildfly, agrupando-os e assim permitindo compartilhar configurações comuns entre eles. Além de compartilhar configurações, é possível também através de um único console de gerenciamento iniciar ou parar instâncias (ou grupos inteiros), verificar seu status e estatísticas de cada subsystem etc. O gerenciamento das instâncias é coordenado pelo Domain Controller, tendo várias instâncias (JVMs) por Host e o controle total do ciclo de vida dos servidores via Host Controller. Para iniciar no modo domain é executado o script JBOSS_HOME/domain.sh no Linux ou JBOSS_HOME/domain.bat no Windows.

Quando o domain.sh (ou o domain.bat) é executado em um host, um processo conhecido como Host Controller é lançado, o qual é responsável pelo gerenciamento do servidor. Ele não lida diretamente com a carga de aplicações no servidor, sendo responsável por iniciar e terminar os processos que rodam em cada servidor, interagindo com o Domain Controller para ajudar no gerenciamento.

Por padrão, cada Host Controller lê sua configuração no arquivo domain/configuration/host.xml, que possui informações de configuração específicas do host. Abaixo podemos ver uma figura com quatro hosts, um Domain Controller e três Host Controllers, sendo que cada Host Controller pode ter um ou mais servidores.



A figura acima mostra uma configuração em modo domain, sendo que todas as configurações e o gerenciamento são realizados de forma centralizada, no Domain Controller. Todas as configurações realizadas para o Domain são replicadas nas instâncias JBoss. Adicionar um novo grupo de servidores, configurar logs, criar data sources, alterar portas, entre outras coisas, tudo isso é feito no domain.xml do arquivo de configuração do Domain Controller (master).

Características como *clustering*, *high availability* (HA), *fail-over* e outros recursos do JEE estão disponíveis nos dois modos. Domain Controller é quem controla o gerenciamento do *domain*. Nele estão as configurações que são compartilhadas entre as instâncias que estão nesse *domain*, e a política de gerenciamento de todos os servidores. O Domain Controller é basicamente um processo Host Controller que dependendo da arquitetura se torna o Domain Controller.

Foi reduzida a necessidade de editar arquivos de configuração XML manualmente. O gerenciamento da segurança é realizado de forma simplificada, inclusive para domínios de segurança. O diretório "modules" centraliza os módulos do servidor de aplicações, em vez do diretório "lib". Os diretórios "domain" e "standalone" possuem os arquivos de configurações para *deploys* (implantações) em modo domain e standalone.

O mecanismo de carga de classe é modular, então os módulos são carregados e descarregados sob demanda, o que propicia uma melhora na segurança, além de menores tempos para iniciar ou reiniciar o servidor. Duas bibliotecas iguais de versões diferentes podem conviver no mesmo servidor.

Instalação, Configuração e Implantação

Para a instalação, existe a opção de baixar o JBoss/Wildfly em binário, em arquivo zip ou com um instalador. A maneira mais rápida é fazer o download do binário, e descompactar em um diretório. Há alguns riscos que devem ser evitados na instalação, para evitar falhas de segurança.

Assim, após a instalação, há a necessidade de ajustar e personalizar a configuração do servidor, antes de colocar o servidor em produção.

O pré-requisito para instalar o JBoss é o Java Development Kit (JDK) ou Java Runtime Environment (JRE) instalado. O download do JBoss/Wildfly pode ser realizado no site da RedHat. Após o download, os arquivos de instalação devem ser colocados na pasta adequada, para iniciar a instalação.

Após a instalação, é necessário configurar o JBoss/Wildfly como serviço. Não é recomendado iniciá-lo, e deixá-lo configurado com o usuário root, pois isto pode comprometer a segurança de toda a plataforma, já que a plataforma Java oferece APIs para execução de códigos nativos do sistema operacional e mecanismos de gerenciamento remoto.

O ideal é que no Linux seja criado um usuário com privilégios adequados para iniciar o serviço do JBoss, e no Windows seja criado um usuário com poderes administrativos, mas com privilégios reduzidos. Como as questões de concurso geralmente abordam comandos (para a configuração ou para a administração), vamos dar uma olhada abaixo.

Após a instalação, é necessário criar um grupo:

```
# groupadd jboss
```

Para adicionar o usuário jboss no grupo jboss:

```
# useradd -s /bin/bash -d /home/jboss -m -g jboss jboss
```

Na sequência, a criação da estrutura de diretórios para armazenar o JBoss, atribuindo o dono e o grupo “jboss” a essa estrutura:

```
# mkdir /EAP_HOME/jboss

# chown jboss:jboss /EAP_HOME/jboss

# su jboss
```

Depois é necessário configurar a senha para que o usuário a utilize na conexão entre o Host Controller e o Domain Controller. Dando continuidade, agora vamos criar os perfis para o domain controller (master) e para os host controllers (slave), baseados no modo Domain e renomeá-los:

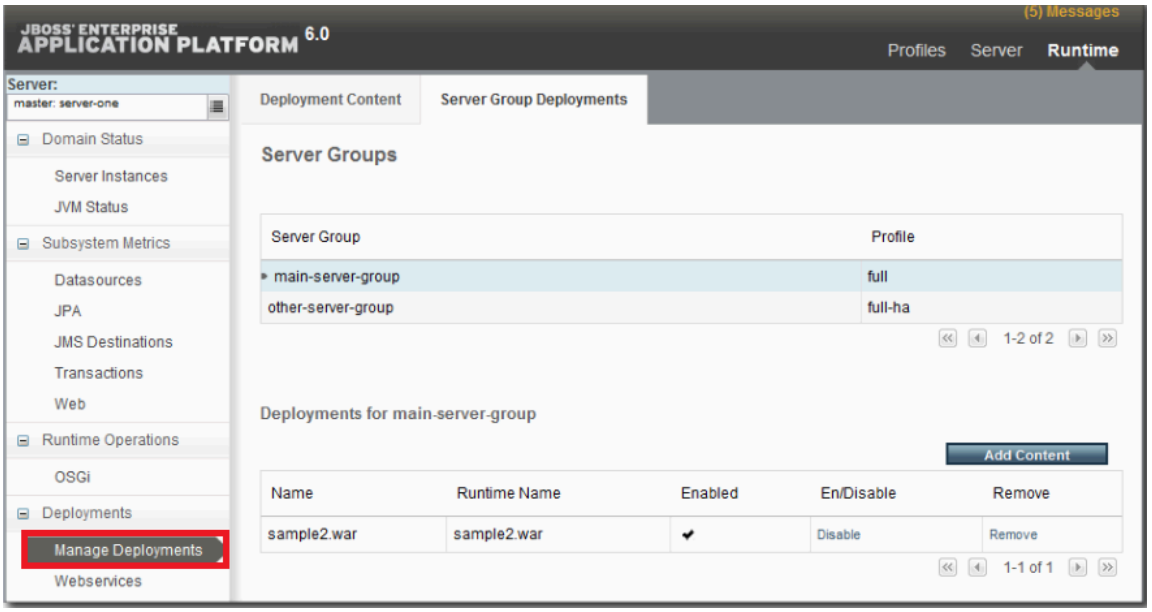
```
# cp -Rap /EAP_HOME/domain /EAP_HOME/master

# cp -Rap /EAP_HOME/domain /EAP_HOME/slave01
```

Após a definição do modo (standalone ou domain), é necessário definir um perfil para o JBoss/Wildfly, que é um conjunto de tecnologias ou subsistemas (subsystems) que serão utilizados na aplicação. Essa definição varia conforme os requisitos de cada sistema. O JBoss possui 4 perfis (profiles) por padrão: default, full, ha, full-ha.

Depois de concluída a configuração inicial, as aplicações devem ser "colocadas" no JBoss, processo conhecido como *deploy* (implantação).

O *deploy* varia conforme o tipo de perfil (*profile*) definido para o JBoss/Wildfly. Vamos ver como é feito o *deploy* de uma aplicação no JBoss EAP. Primeiro, devemos acessar o console Web na URL http://x.x.x.x:9990/console/ (onde x.x.x.x é o endereço IP do servidor). Note que a porta utilizada pelo servidor é a 9990, mas roda a aplicação na 8080. Ou seja, o JBoss/Wildfly roda em uma porta e a aplicação em outra.

Em seguida devem ser inseridos o usuário e a senha, aqueles definidos na instalação. Para realizar um *deploy* e o gerenciamento das instâncias, pode ser utilizado o console:



O *deploy* também pode ser realizado através da linha de comando, o que propicia maior flexibilidade e facilidade na criação de *scripts* de administração. Abaixo um exemplo de *deploy* para todos os servidores:

```
# deploy /path/to/test-application.war --all -server-groups
```

Como já vimos, servidores de aplicação são a interface entre o componente e o sistema operacional específico que o suporta. Antes do componente ser executado no servidor, ele precisa ser montado em uma estrutura que o container possa entender e executar. No JBoss/Wildfly, a estrutura de empacotamento é definida na especificação JSR 088 Java EE Application Deployment Specification. Os formatos padrão são WAR, JAR e EAR.

Estrutura de Diretórios

Após a instalação, o JBoss/Wildfly cria uma estrutura de diretórios que deve ser conhecida, pois é necessária para a administração.

- bin/: contém os scripts de inicialização do JBoss EAP, no Linux e no Windows;
- appclient/: contém detalhes de configuração do container da aplicação cliente;

- modules/: contém módulos dinamicamente carregados pelo JBoss EAP quando requeridos pelos serviços;
- standalone/: arquivos de configuração, conteúdo do deploy, e outras áreas utilizáveis quando o JBoss/Wildfly executa como um servidor standalone;
- domain/: contém arquivos de configuração, conteúdo do deploy e outras áreas utilizáveis quando JBoss/Wildfly executa como domínio gerenciado (domain);
- docs/: possui diversos tipos de arquivos como exemplos de configuração, exemplos se como executar o Wildfly como serviço, licenças e outros arquivos que ajudam a aprender mais sobre o Wildfly;
- welcome-content/: diretório de uso interno do servidor que não deve ser modificado por usuários finais, possui páginas de boas vindas e páginas de erros;
- jboss-modules.jar: mecanismo de carga dos módulos.

Além da estrutura de diretórios apresentada, o JBoss/Wildfly cria automaticamente alguns caminhos (*paths*) padrão:

- jboss.home: diretório root do JBoss EAP;
- user.home: diretório de usuário comum;
- user.dir: diretório de trabalho do usuário atual;
- java.home: diretório de instalação do Java;
- jboss.server.base.dir: diretório root de uma instância de um servidor;
- jboss.server.data.dir: diretório que o servidor usa para persistência de dados no storage;
- jboss.server.log.dir: diretório que o servidor usa no armazenamento de logs;
- jboss.server.tmp.dir: diretório de arquivos temporários;
- jboss.domain.servers.dir: diretório no qual um host controller cria a área de trabalho de uma instância em um domínio gerenciado.

Outros conceitos

JGroups: Biblioteca de comunicação em grupo em Java utilizada por diversas aplicações distribuídas para comunicação entre diferentes nós em uma rede. Essa biblioteca oferece suporte à comunicação confiável e escalável em grupos de processos. Ao utilizar o JGroups com o JBoss, as aplicações podem aproveitar esses recursos para criar ambientes distribuídos robustos e escaláveis.

Arquillian: Plataforma de teste para aplicações JEE que simplifica a execução de testes de integração e funcionalidade em containers do JEE. Ferramenta projetada para ajudar os desenvolvedores a realizar testes integrados em ambientes mais próximos da produção, onde componentes como EJBs, CDI (Contexts and Dependency Injection), JPA (Java Persistence API), entre outros, são executados em um ambiente Java EE real.

Operação (Resumo)

De acordo com o que vimos na aula, um resumo da operação do JBoss é o seguinte:

- Configuração do JBoss em ambientes distintos (produção, desenvolvimento e homologação) com arquivos de configuração apropriados (standalone.xml ou domain.xml);

- Para a implantação de aplicações, pode-se utilizar a console de administração ou scripts CLI para fazer o deploy de aplicações em arquivos .war, .ear, .jar etc.;

- Gerenciamento de modos: Standalone (um único servidor) ou Domain Mode (gerenciamento de múltiplos servidores de forma centralizada);

- A configuração e o monitoramento dos logs (standalone ou domain) são fundamentais para a resolução de problemas e auditoria.

Monitoramento

Para o monitoramento de métricas de CPU, memória, e desempenho de aplicativos existem ferramentas, como Prometheus, Grafana, Nagios, ou até mesmo o JConsole (ferramenta padrão da JDK). Podemos destacar, também:

- MBeans e JMX: O JBoss utiliza Java Management Extensions (JMX) para expor métricas e operações administrativas através dos MBeans. Estes permitem observar a utilização de CPU, memória e status de aplicações;

- Alertas e Notificações: É importante configurar alertas para eventos específicos, como falhas no servidor, excesso de uso de CPU/memória etc. Com esses alertas é possível uma atuação proativa facilitada.

Administração (Resumo)

Conforme vimos nesta aula, vamos destacar alguns pontos importantes para a administração do JBoss:

- Console de Administração: Com a interface Web de administração é possível gerenciar deploys, configurar fontes de dados, ajustar a configuração de segurança, gerenciar clusters, entre outras atividades;

- CLI (*Command Line Interface*): É ideal para operações automatizadas, como scripts de deploy em diversos ambientes ou atualizações em grande escala;

- Gestão de Clusters: No modo Domain, é possível gerenciar diversas instâncias de servidor e balancear a carga entre elas;

- Backup e Recuperação: É importante determinar rotinas de backup dos arquivos de configuração e bases de dados associadas, pois são essenciais para a recuperação de desastres.

Ajustes de Desempenho

Com a intenção de manter o servidor JBoss/WildFly estável, seguro e com alto desempenho para aplicações Java empresariais, alguns processos e ferramentas são:

- Ajuste de JVM: Otimização da configuração da JVM (*Java Virtual Machine*) para alocação de memória, como o heap e o permgen/metaspace;

- Ajuste de Datasources (fontes de dados): Configuração de pools de conexão para limitar o uso de recursos e evitar vazamento de conexões, especialmente para conexões com banco de dados;

- Cache e Sessão: Utilização de cache para melhorar o desempenho em clusters, especialmente em aplicações que dependem de sessões de usuário;

- Perfis de Desempenho (Performance): O JBoss/Wildfly permite a configuração de diferentes perfis de desempenho, como "High Availability" para clusters ou "Full" para ambientes que exigem alta carga de processamento.

Referências Bibliográficas

O que é JBoss. Disponível em <https://4linux.com.br/o-que-e-jboss/>.

Red Hat JBoss Enterprise Application Platform. Disponível em <https://www.redhat.com/pt-br/technologies/jboss-middleware/application-platform>.

WildFly. A powerful, modular, & lightweight application server that helps you build amazing applications. Disponível em <https://www.wildfly.org/>.

QUESTÕES COMENTADAS - SERVIDOR DE APLICAÇÃO JBOSS/WILDFLY - MULTIBANCAS

1. (IBFC/EBSERH - 2016) "JBoss é um servidor _____ de código fonte _____ baseado na plataforma _____ e implementado na linguagem de programação _____". Assinale a alternativa que completa correta e respectivamente as lacunas:

A) de arquivos - aberto - JEE - JavaScript

B) de aplicação - fechado - JSE - Java

C) de arquivos - fechado - JEE - JavaScript

D) de aplicação - aberto - JEE - Java

E) de impressão - aberto - JSE - JavaScript

Comentários:

Conceitos básicos que vimos logo no primeiro parágrafo:

JBoss é um servidor de aplicação de código fonte aberto baseado na plataforma JEE, é implementado completamente na linguagem de programação Java, podendo ser utilizado em qualquer sistema operacional que suporte essa linguagem. O JBoss Application Server utiliza o arquivo standalone.bat (ou standalone.sh) para prover a sua inicialização. Portanto, a alternativa D está correta e é o gabarito da questão.

Gabarito: Letra D

2. (FUNCAB/CREA-AC - 2016) Um administrador de rede instalou o Jboss AS 7 no modo domain. Nesse caso, um dos processos principais, que coordena as instâncias e distribui o arquivo implantado para todas as instâncias do domínio, é denominado:

A) application process.

B) host controller.

C) JVM process.

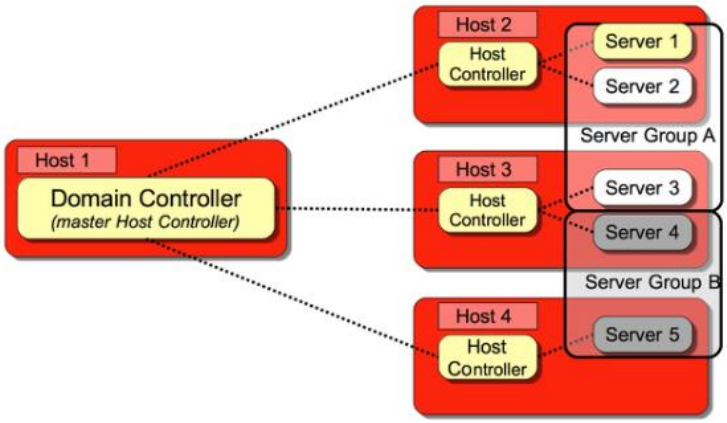
D) system controller.

E) standalone controler.

Comentários:

Quando o domain.sh (ou o domain.bat) é executado em um host, um processo conhecido como Host Controller é lançado, o qual é responsável pelo gerenciamento do servidor. Ele não lida diretamente com a carga de aplicações no servidor, sendo responsável por iniciar e terminar os processos que rodam em cada servidor, interagindo com o Domain Controller para ajudar no gerenciamento.

Por padrão, cada Host Controller lê sua configuração no arquivo domain/configuration/host.xml, que possui informações de configuração específicas do host. Abaixo podemos ver uma figura com quatro hosts, um Domain Controller e três Host Controllers, sendo que cada Host Controller pode ter um ou mais servidores.



Portanto, a alternativa B está correta e é o gabarito da questão.

Gabarito: Letra B

3. (FCC/TRT11 - 2017) Após instalar o servidor JBoss AS 5 para Windows, deve-se entrar na pasta $JBOSS_HOME/bin e digitar run.bat para iniciá-lo. Com o servidor iniciado, para acessar a área de login do JBoss AS Administration Console deve-se digitar,

A) na linha de endereço do navegador, http://localhost:8080/admin-console.

B) em linha de comando, jboss -a console.

C) na linha de endereço do navegador, http://localhost:8084/settings.

D) em linha de comando, jboss -a mode=console.

E) na linha de endereço do navegador, http://localhost:80/server-console.

Comentários:

O deploy varia conforme o tipo de perfil (profile) definido para o JBoss/Wildfly. Vamos ver como é feito o deploy de uma aplicação no JBoss EAP. Primeiro, devemos acessar o console Web na URL http://x.x.x.x:9990/console/ (onde x.x.x.x é o endereço IP do servidor). Note que a porta utilizada pelo servidor é a 9990, mas roda a aplicação na 8080. Ou seja, o JBoss/Wildfly roda em uma porta e a aplicação em outra.

Note que foi cobrada uma versão mais antiga (AS 5) e teve uma leve variada o nome, mas o que vale é focar na porta 8080, que é a chave da questão!

Portanto, a alternativa A está correta e é o gabarito da questão.

Gabarito: Letra A

4. (IBFC/EMBASA - 2017) Para que o servidor JBOSS Application Server 7 possa ser plenamente executado, existe a necessidade que esteja previamente instalado e configurado o:

A) JMF

B) JCE

C) JXL

D) JDK

Comentários:

JBoss é um servidor de aplicação de código fonte aberto baseado na plataforma JEE, é implementado completamente na linguagem de programação Java, podendo ser utilizado em qualquer sistema operacional que suporte essa linguagem.

Para suportar a linguagem Java, deve haver instalado o JDK (Java Development Kit), que é um ambiente utilizado para o desenvolvimento de softwares em Java. O JDK inclui o JRE (Java Runtime Environment), um interpretador/carregador, um compilador (javac), entre outros componentes.

Pode ter apenas o JRE instalado, que também funciona! Mas nas alternativas só encontramos o JDK!

Portanto, a alternativa D está correta e é o gabarito da questão.

Gabarito: Letra D

5. (FGV/Câmara de Salvador-BA - 2018) No âmbito do JBoss AS 7.x, os modos de operação disponíveis são denominados:

A) Domain e Standalone;

B) Elevated e Standard;

C) Local e Remote;

D) Monouser e Multiuser;

E) Open e Authenticated.

Comentários:

Modo Standalone: é o modo tradicional das versões anteriores. Basicamente implica em ter uma instalação diferente (ou um diretório standalone diferente) para cada instância de Wildfly. Ou seja, para cada Wildfly em execução no seu ambiente é necessário alterar seus próprios arquivo de configuração, suas próprias opções de execução para JVM etc. A inicialização nesse modo é realizada ao iniciar o script JBOSS_HOME/standalone.sh (no Linux) ou JBOSS_HOME/standalone.bat (no Windows).

Modo Domain: é o modo que foi introduzido no JBoss AS 7, permitindo gerenciar um conjunto de instâncias Wildfly, agrupando-os e assim permitindo compartilhar configurações comuns entre eles. Além de compartilhar configurações, é possível também através de um único console de gerenciamento iniciar ou parar instâncias (ou grupos inteiros), verificar seu status e estatísticas de cada subsystem etc. O gerenciamento das instâncias é coordenado pelo Domain Controller, tendo várias instâncias (JVMs) por Host e o controle total do ciclo de vida dos servidores via Host Controller. Para iniciar no modo domain é executado o script JBOSS_HOME/domain.sh no Linux ou JBOSS_HOME/domain.bat no Windows.

Portanto, a alternativa A está correta e é o gabarito da questão.

Gabarito: Letra A

6. (FCC/MPE-PE - 2018) Em uma instalação padrão do JBoss Application Server (AS) 7, o diretório que contém a página de boas-vindas do AS é o

A) standalone.

B) welcome-content.

C) appclient.

D) server-welcome.

E) bundles.

Comentários:

Após a instalação, o JBoss/Wildfly cria uma estrutura de diretórios que deve ser conhecida, pois é necessária para a administração.

- bin/: contém os scripts de inicialização do JBoss EAP, no Linux e no Windows;
- appclient/: contém detalhes de configuração do container da aplicação cliente;
- modules/: contém módulos dinamicamente carregados pelo JBoss EAP quando requeridos pelos serviços;
- standalone/: arquivos de configuração, conteúdo do deploy, e outras áreas utilizáveis quando o JBoss/Wildfly executa como um servidor standalone;
- domain/: contém arquivos de configuração, conteúdo do deploy e outras áreas utilizáveis quando JBoss/Wildfly executa como domínio gerenciado (domain);
- docs/: possui diversos tipos de arquivos como exemplos de configuração, exemplos se como executar o Wildfly como serviço, licenças e outros arquivos que ajudam a aprender mais sobre o Wildfly;
- welcome-content/: diretório de uso interno do servidor que não deve ser modificado por usuários finais, possui páginas de boas vindas e páginas de erros;
- jboss-modules.jar: mecanismo de carga dos módulos.

Portanto, a alternativa B está correta e é o gabarito da questão.

Gabarito: Letra B

7. (FCC/CLDF - 2018) Considere que o servidor de aplicações JBoss AS 7 está instalado e configurado em modo padrão em um computador com sistema operacional Windows 10. Para testar se o servidor JBoss está funcionando, utilizando um navegador, deve-se digitar o URL

A) server://127.0.0.1:0800

B) ftp://localhost:0800

C) http://localhost:8080

D) server://localhost:80

E) http://127.0.0.1:80

Comentários:

O deploy varia conforme o tipo de perfil (profile) definido para o JBoss/Wildfly. Vamos ver como é feito o deploy de uma aplicação no JBoss EAP. Primeiro, devemos acessar o console Web na URL http://x.x.x.x:9990/console/ (onde x.x.x.x é o endereço IP do servidor). Note que a porta

utilizada pelo servidor é a 9990, mas roda a aplicação na 8080. Ou seja, o JBoss/Wildfly roda em uma porta e a aplicação em outra. Portanto, a alternativa C está correta e é o gabarito da questão.

Gabarito: Letra C

Lista de Questões - Servidor de Aplicação JBoss/Wildfly - Multibancas

1. (IBFC/EBSERH - 2016) "JBoss é um servidor _____ de código fonte _____ baseado na plataforma _____ e implementado na linguagem de programação _____". Assinale a alternativa que completa correta e respectivamente as lacunas:

A) de arquivos - aberto - JEE - JavaScript

B) de aplicação - fechado - JSE - Java

C) de arquivos - fechado - JEE - JavaScript

D) de aplicação - aberto - JEE - Java

E) de impressão - aberto - JSE - JavaScript

2. (FUNCAB/CREA-AC - 2016) Um administrador de rede instalou o Jboss AS 7 no modo domain. Nesse caso, um dos processos principais, que coordena as instâncias e distribui o arquivo implantado para todas as instâncias do domínio, é denominado:

A) application process.

B) host controller.

C) JVM process.

D) system controller.

E) standalone controler.

3. (FCC/TRT11 - 2017) Após instalar o servidor JBoss AS 5 para Windows, deve-se entrar na pasta $JBOSS_HOME/bin e digitar run.bat para iniciá-lo. Com o servidor iniciado, para acessar a área de login do JBoss AS Administration Console deve-se digitar,

A) na linha de endereço do navegador, http://localhost:8080/admin-console.

B) em linha de comando, jboss -a console.

C) na linha de endereço do navegador, http://localhost:8084/settings.

D) em linha de comando, jboss -a mode=console.

E) na linha de endereço do navegador, http://localhost:80/server-console.

4. (IBFC/EMBASA - 2017) Para que o servidor JBOSS Application Server 7 possa ser plenamente executado, existe a necessidade que esteja previamente instalado e configurado o:

A) JMF

B) JCE

C) JXL

D) JDK

5. (FGV/Câmara de Salvador-BA - 2018) No âmbito do JBoss AS 7.x, os modos de operação disponíveis são denominados:

A) Domain e Standalone;

B) Elevated e Standard;

C) Local e Remote;

D) Monouser e Multiuser;

E) Open e Authenticated.

6. (FCC/MPE-PE - 2018) Em uma instalação padrão do JBoss Application Server (AS) 7, o diretório que contém a página de boas-vindas do AS é o

A) standalone.

B) welcome-content.

C) appclient.

D) server-welcome.

E) bundles.

7. (FCC/CLDF - 2018) Considere que o servidor de aplicações JBoss AS 7 está instalado e configurado em modo padrão em um computador com sistema operacional Windows 10. Para testar se o servidor JBoss está funcionando, utilizando um navegador, deve-se digitar o URL

A) server://127.0.0.1:0800

B) ftp://localhost:0800

C) http://localhost:8080

D) server://localhost:80

E) http://127.0.0.1:80

GABARITO

1- D
2- B
3- A
4- D
5- A
6- B
7- C

Oracle WebLogic

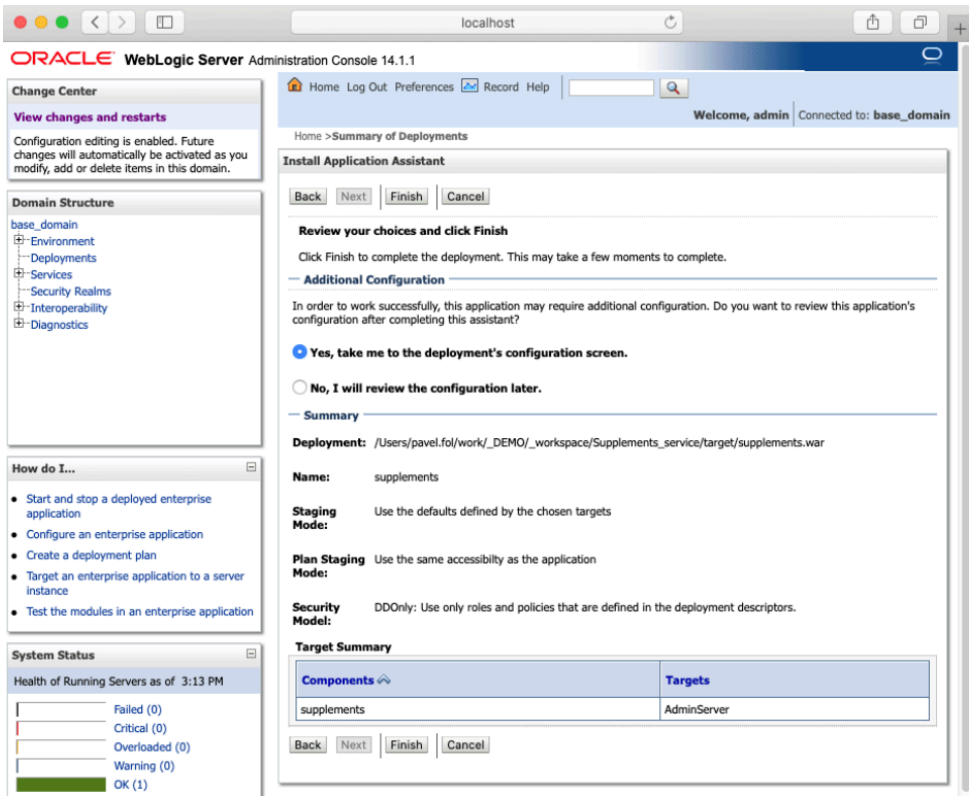
Segundo o sítio do fabricante, o Oracle WebLogic Server é "uma plataforma unificada e extensível para desenvolver, implementar e executar aplicações corporativas, como Java, on-premises e na nuvem. O Weblogic Server oferece uma implementação robusta, madura e escalável do Java Enterprise Edition (EE) e Jakarta EE.". Oferece um ambiente de alta disponibilidade, escalabilidade e segurança, adequado para aplicações empresariais de grande porte. Na sequência veremos um resumo dos principais aspectos dessa plataforma.

Operação

Vamos ver alguns detalhes da operação:

- Configuração de Ambientes: Pode ser configurado em ambientes distintos (produção, homologação, desenvolvimento) com perfis personalizados de configuração;

- Implantação de Aplicações: Permite implantação (*deploy*) de aplicações (arquivos .ear, .war, .jar) por meio da "Admin Console", scripts WLST (*WebLogic Scripting Tool*), ou diretamente pela linha de comando (CLI - *Command Line Interface*). Abaixo um exemplo (imagem) do "Admin Console";



- Domínios e Clusters: Os domínios WebLogic possuem um ou mais servidores WebLogic agrupados. Eles podem ser configurados para suportar clusters, distribuindo a carga entre os nós, proporcionando alta disponibilidade;

- Logs e Auditoria: O WebLogic realiza o registro de logs operacionais, logs de erro e logs de auditoria. Esses logs ajudam a rastrear a operação do servidor e são essenciais para a resolução de problemas.

Monitoramento

Em relação ao monitoramento existem algumas ferramentas:

- Painel de Administração (*Admin Console*): A interface gráfica de administração do WebLogic (imagem já mostrada anteriormente) permite monitorar o status do servidor, estatísticas de uso de memória e CPU, desempenho de aplicações e status de fontes de dados (*datasources*);

- WLDF (*WebLogic Diagnostic Framework*): Framework que permite monitorar métricas detalhadas e gerar alertas baseados em regras personalizadas para monitorar atividades específicas, incluindo gargalos e falhas;

- JMX e SNMP: Métricas de desempenho e monitoramento através de JMX (*Java Management Extensions*) e há o suporte para SNMP (*Simple Network Management Protocol*) para a integração com ferramentas de monitoramento;

- Ferramentas Externas: Algumas ferramentas como Oracle Enterprise Manager, Prometheus, Grafana e Nagios também são utilizadas para monitorar o WebLogic em tempo real.

Administração

Vamos ver alguns pontos relacionados à administração do WebLogic:

- Admin Console: A interface Admin Console é sem dúvidas a principal ferramenta de administração gráfica do WebLogic;

- Configuração de Clusters e Balanceamento de Carga: O WebLogic permite configurar *clusters* de servidores para o balanceamento de carga e *failover*, essencial para aplicações de alta disponibilidade (HA - *High Availability*);

- Gerenciamento de Segurança: Inclui configuração de autenticação, controle de acesso baseado em funções (RBAC - *Role Based Access Control*), configuração de SSL/TLS e uso de usuários e grupos do sistema. Existe integração com provedores de identidade, como o LDAP (*Lightweight Directory Access Protocol*);

- *Scripts* de Automação: Utilizando WLST (*WebLogic Scripting Tool*), os administradores podem automatizar operações de administração rotineiras, como *deploys*, configurações de servidores e backups. A WLST é uma ferramenta de linha de comando e *scripting* poderosa, essencial para automação e gerenciamento em larga escala.

Ajustes de Desempenho

Os elementos abaixo tornam o WebLogic Server altamente eficiente e confiável para executar aplicações corporativas complexas, além de serem essenciais para manter o ambiente estável e de alto desempenho:

- Configuração da JVM: Ajustes de parâmetros da JVM são fundamentais para o desempenho do WebLogic. Importante configurar heap, metaspace e garbage collection, de acordo com a carga de trabalho esperada;

- Ajuste de Fontes de Dados (*Datasources*): *Datasources* devem ser configurados com atenção para evitar exaustão de conexões. Parâmetros como "Tamanho Máximo do Pool", "Timeouts de Inatividade", e "Testes de Conexão" são importantes para a estabilidade e bom desempenho do banco de dados;

- Ajuste (*Tuning*) de *Threads*: Ajustar a quantidade de *threads* do "Thread Pool" ajuda a otimizar a resposta do servidor e evitar gargalos;

- Cache e Sessão: A utilização de mecanismos de cache reduz o tempo de resposta. Deve-se configurar a replicação de sessão em *clusters* para garantir que o estado da sessão seja preservado;

- Ajustes de Rede: Configurações como "Keep-Alive", "Buffering", e "Timeouts" são essenciais para otimizar a conectividade do servidor com aplicações externas.

Referências Bibliográficas

Java Basics: What Is WebLogic? Disponível em <https://www.jrebel.com/blog/weblogic>

Oracle WebLogic Server. Disponível em <https://www.oracle.com/br/java/weblogic>.

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.