

Grupo Handbook

Handbook de Questões de TI

comentadas para CONCURSOS

Além do gabarito

Volume 6

*Suporte de Infraestrutura
Parte 1
Fundação Cesgranrio*

Prefácio

Este é o volume 6 da série *Handbook de Questões de TI Comentadas para Concursos – Além do Gabarito*, que traz para você uma coletânea especial de questões da área de suporte de infraestrutura selecionadas de provas organizadas pela Fundação Cesgranrio.

Entre as provas das quais as questões foram selecionadas, estão a primeira prova para Analista de Suporte para o BNDES em 2008, a prova para Analista de Suporte de Infraestrutura para a Petrobras aplicada em 2008, e, também, a prova para Analista de Sistemas – Suporte para o IBGE aplicada em 2010.

Notoriamente, a Fundação Cesgranrio vem sendo a responsável pela organização de concursos de TI de alta concorrência e visibilidade, como os do BNDES, e os da Petrobras e suas subsidiárias. Mais recentemente, a Fundação Cesgranrio também foi a escolhida para realizar os concursos do Banco Central e de outras instituições do ciclo de gestão do Governo Federal.

Uma interessante característica das provas aplicadas pela Fundação Cesgranrio é o fato das questões possuírem pesos diferentes, o que exige dos candidatos ainda mais sagacidade. Além disso, é comum que sejam cobrados temas recentes de TI, como forma de selecionar profissionais atualizados e dinâmicos.

Por isso, o Grupo Handbook de TI preparou para você uma edição especial com questões da área de suporte de infraestrutura, abordando temas clássicos e modernos da computação. Este volume é especialmente útil para todos que desejam prestar concursos na área de suporte, em especial, aqueles organizados pela Fundação Cesgranrio.

Bons estudos,

Grupo Handbook de TI

Direitos Autorais

Este material é registrado no Escritório de Direitos Autorais (EDA) da Fundação Biblioteca Nacional. Todos os direitos autorais referentes a esta obra são reservados exclusivamente aos seus autores.

Os autores deste material não proíbem seu compartilhamento entre amigos e colegas próximos de estudo. Contudo, a reprodução, parcial ou integral, e a disseminação deste material de forma indiscriminada através de qualquer meio, inclusive na Internet, extrapolam os limites da colaboração. Essa prática desincentiva o lançamento de novos produtos e enfraquece a comunidade concurseira Handbook de TI.

A série *Handbook de Questões de TI Comentadas para Concursos – Além do Gabarito* é uma produção independente e contamos com você para mantê-la sempre viva.

Grupo Handbook de TI

Canais de Comunicação

O Grupo Handbook de TI disponibiliza diversos canais de comunicação para os concurseiros de TI.

Loja Handbook de TI

Acesse a nossa loja virtual em <http://www.handbookdeti.com.br>

Serviço de Atendimento

Comunique-se diretamente conosco através do e-mail faleconosco@handbookdeti.com.br

Twitter do Handbook de TI

Acompanhe de perto promoções e lançamentos de produtos pelo nosso Twitter <http://twitter.com/handbookdeti>

1. **Assuntos relacionados:** *Servidor de Aplicações, Java, J2EE, Máquina Virtual Java (JVM), Heap,*
Banca: *Cesgranrio*
Instituição: *BNDES*
Cargo: *Analista de Sistemas - Suporte*
Ano: *2008*
Questão: *31*

Um servidor Linux, que roda, exclusivamente, um servidor de aplicação Java EE, possui 2 GB de memória RAM e 1 CPU. A única aplicação em execução atinge, em momentos de pico, 50 usuários simultâneos. Para que essa aplicação tenha um desempenho adequado, o tamanho máximo da Heap da JVM pode ser configurado para

- (a). 100 threads.
- (b). 32 MB.
- (c). 60 threads.
- (d). 2 GB.
- (e). 512 MB.

Solução:

O Java EE (Enterprise Edition) é uma plataforma de desenvolvimento de sistemas em Java. A plataforma inicialmente era conhecida por Java 2 Platform Enterprise Edition (J2EE), até ter seu nome trocado para Java EE, o que ocorreu na versão 5.

A plataforma JEE oferece uma série de componentes e funcionalidades que permitem a implementação de software Java distribuído. Uma descrição sucinta da destinação e das características gerais da plataforma JEE pode ser encontrada na Wikipedia:

A Plataforma Java (Enterprise Edition) difere-se da Plataforma Java Standard Edition (Java SE) pela adição de bibliotecas que fornecem funcionalidade para implementar software Java distribuído, tolerante a falhas e multi-camada, baseada amplamente em componentes modulares executando em um servidor de aplicações. Ela é voltada para aplicações multi-camadas, baseadas em componentes que são executados em um servidor de aplicações... Ela contém bibliotecas desenvolvidas para o acesso a base de dados, RPC, CORBA, etc. Devido a essas características a plataforma é utilizada principalmente para o desenvolvimento de aplicações corporativas.

Os servidores de aplicação mencionados na descrição acima, em linhas gerais, são softwares que provêem um ambiente onde as aplicações podem executar.

Uma aplicação Java usualmente é dividida em 2 partes: uma parte que executa no cliente e uma outra parte que executa no servidor. O servidor em si é dividido em diferentes *containers* que oferecem diferentes serviços às aplicações. Entre os serviços mais comuns oferecidos pelos *containers* estão a execução da lógica de negócio, serviços de mensagens, serviços de gerenciamento de conexões a bancos de dados, serviços de segurança etc.

Entendido o conceito de servidores de aplicação, agora vamos falar um pouco sobre JVM (Java Virtual Machines).

Quando um código Java é compilado, ele dá origem a um programa codificado nos chamados *bytecodes*. Os *bytecodes* são uma espécie de código intermediário, que só pode ser executado por uma JVM. Uma JVM, por sua vez, é um programa que converte os *bytecodes* em código executável de máquina, que pode então ser executado pela máquina física.

Em última instância, são os conceitos de *bytecodes* que garantem a portabilidade dos códigos Java. Independente do sistema operacional ou plataforma física onde tenha sido desenvolvido o programa Java, os *bytecodes* sempre serão os mesmos. Com isso, para executar o programa Java em outro sistema, basta que se tenha uma JVM específica, capaz de converter os *bytecodes* para o código de máquina apropriado.

Agora, vamos ao que realmente interessa para alcançarmos a resposta da questão.

A *heap* da JVM é uma área de memória onde todos os objetos das aplicações que estão sendo executadas pela JVM residem. Além dos objetos criados nas aplicações, a *heap* ainda possui uma área de memória reservada para outras funções da JVM. Portanto, para que as aplicações executem com desempenho adequado, é necessário configurar bem o tamanho da *heap*.

As alternativas A e C estão erradas, simplesmente, porque o tamanho da *heap* não é definido em termos do número de threads, mas sim em termos de quantidade de memória. Já a alternativa D está errada pois não faz sentido alocar toda a memória do sistema (no caso, 2GB) exclusivamente para a JVM. O sistema operacional sequer irá permitir a realização desta operação, visto que boa parte da memória já estará sendo utilizada pelo próprio sistema operacional.

Com isto, nos restam as alternativas B (32 MB) e D (512 MB). O enunciado da questão diz que o servidor possui 1 CPU, e que, em momentos de pico, a aplicação em execução atinge 50 usuários simultâneos. O fato de o servidor possuir apenas 1 CPU implica dizer que, em momentos de alta utilização da aplicação, haverá muitas trocas de contexto, ou seja, muitas alternâncias entre os processos em execução.

Em princípio, a alteração da quantidade de memória não irá afetar a quantidade de trocas de contexto, porém irá permitir que as múltiplas threads (que serão criadas para atender uma grande quantidade de usuários simultâneos) estejam sempre na memória principal, o que pode conferir maior desempenho à aplicação.

Como o enunciado não presta informações sobre o consumo de memória das threads individuais que são geradas para atender cada um dos usuários simultâneos, e levando ainda em consideração que o tamanho default da *heap* (ou seja, um tamanho que atende os requisitos de desempenho de uma aplicação média) da JVM varia entre 16MB a 64MB, podemos afirmar que, para maximizar o desempenho da aplicação, o ideal seria configurar o tamanho máximo da *heap* para 512MB.

Portanto, a resposta da questão é a alternativa E.

2. Assuntos relacionados: *Sistemas Operacionais, Linux, Shell,***Banca:** *Cesgranrio***Instituição:** *BNDES***Cargo:** *Analista de Sistemas - Suporte***Ano:** *2008***Questão:** *34*

Um servidor Linux de produção apresenta, esporadicamente, um erro conhecido, no qual um determinado processo (cgi.rb) ocupa 99% de CPU indeterminadamente. Você, como operador, é instruído a matar esse processo para que o desempenho do servidor volte à situação normal. Que seqüência de operações deve ser feita?

- (a). Localizar no /proc o processo problemático e executar o comando halt.
- (b). Descobrir o número do processo com ps e executar o comando kill.
- (c). Verificar o id do processo no arquivo /var/log/err e apagá-lo do /proc.
- (d). Reiniciar o servidor e apagar, recursivamente, o /proc.
- (e). Reiniciar o serviço de rede e apagar o /var/run/cgi.rb.pid.

Solução:

O comando utilizado no linux para matar um processo é o comando kill. Este comando envia por padrão a mensagem de TERM ao processo que for passado como parâmetro, informando ao mesmo que ele deve ser finalizado. O parâmetro a ser passado, entretanto, não é o nome do processo e sim seu numero de identificação (pid –process id). A sintaxe do comando neste caso é:

```
kill <pid>
```

```
Exemplo: kill 3568
```

Para obtermos o pid de um processo no linux a partir de seu nome utilizamos o comando ps. Para se obter a lista de todos os processos em execução em um dado momento executamos o comando:

```
ps ax
```

A opção “a” informa que desejamos listar os processos de todos os usuários e não apenas os processos do usuário com que estamos logados. Isto é necessário pois em geral os processos de serviços são executados com a conta de usuários do sistema.

A opção “x” informa que desejamos listar não somente os processos que estejam associados a um terminal, mas também os processos que rodam em background, o que é o caso dos processos que rodam como serviços. O comando ps ax retornaria informações como as mostradas abaixo.

```
PID TTY      STAT   TIME COMMAND
  1  ?        Ss     0:06 init [3]
  2  ?        S<     0:33 [migration/0]
  3  ?        SN     0:03 [ksoftirqd/0]
  4  ?        S<     0:00 [watchdog/0]
  5  ?        S<     0:01 [events/0]
  6  ?        S<     0:00 [khelper]
  7  ?        S<     0:00 [kthread]
  9  ?        S<     0:00 [xenwatch]
 10  ?        S<     0:00 [xenbus]
 15  ?        S<     0:14 [migration/1]
 16  ?        SN     0:02 [ksoftirqd/1]
...
...
...
28458 ?        Ss     0:00 sshd: candidat [priv]
28484 ?        S      0:00 sshd: candidat@pts/0
28485 ?        Ss     0:00 -jailshell
28489 ?        S      0:00 -jailshell
29213 ?        Ss1    0:54 /usr/sbin/clamd
```

Desta forma, devemos primeiramente descobrir o número do processo com o comando “ps” e depois matar o processo utilizando o comando “kill”, conforme indicado na alternativa (B).

Avaliando as demais alternativas vemos que na alternativa (A) é mencionada a pasta /proc. Em sistemas operacionais linux existe uma pasta /proc no sistema de arquivos que é uma representação do estado atual do kernel, isto é, do conteúdo de alguns endereços de memória aonde está situado o kernel do sistema operacional. Sua principal utilidade consiste em ser um meio simples de obter informações.

Apesar de poder ser utilizado para obter detalhes sobre um determinado processo é preciso antes obter seu número de identificação (pid). Desta forma, não é possível localizar no /proc o processo problemático, o que elimina a alternativa (A). A pasta /proc também não pode ser utilizada para matar um processo, o que elimina de as alternativas (C) e (D).

A alternativa (A) ainda menciona o comando halt do linux, que é utilizado para reboatar, paralisar ou desligar o computador, não sendo utilizado para matar um processo específico.

A alternativa (E) menciona um arquivo na pasta /var/run. Na pasta /var/run de sistemas linux alguns processos, ao serem criados, geram um arquivo com extensão pid aonde informam o seu número de identificação (pid). Esta poderia ser uma forma de obter o pid de um serviço. Entretanto, o simples fato de apagar este arquivo não causa a finalização do processo associado e desta forma a alternativa (E) fica eliminada.

3. **Assuntos relacionados:** *Redes de Computadores, Serviços de Rede, Protocolos de Rede, DNS,*

Banca: *Cesgranrio*

Instituição: *BNDES*

Cargo: *Analista de Sistemas - Suporte*

Ano: *2008*

Questão: *36*

Uma pequena empresa disponibiliza um site na Internet em uma infra-estrutura própria. O servidor de DNS apresenta problemas de sobrecarga devido a um grande número de consultas realizadas. Considerando-se que não há mudanças freqüentes de endereços IP e que as consultas, oriundas de usuários legítimos, são relacionadas ao servidor WEB, que alteração pode ser feita na configuração do servidor DNS para reduzir consideravelmente sua sobrecarga?

- (a). Forçar que respostas sejam enviadas sempre via UDP.
- (b). Aumentar o TTL para respostas positivas.
- (c). Colocar o IP do default gateway como uma entrada estática no ARP.
- (d). Eliminar o excesso de registros do tipo A.
- (e). Criar um registro CNAME e PTR para o servidor WEB.

Solução:

Originalmente, o mapeamento de nomes em endereços era todo registrado em um único arquivo, chamado HOSTS.TXT. Esse arquivo era gerenciado por uma entidade chamada NIC (Network Information Center). A distribuição das atualizações desse arquivo era feita via FTP, o que consumia uma banda muito grande. Ao longo do tempo, o perfil do usuário de redes e da Internet mudou muito, ocorrendo um grande crescimento das redes locais, que possuíam suas próprias necessidades de mapear nomes que só faziam sentido em seus ambientes. Da mesma forma, o perfil das aplicações de Internet evoluiu, surgindo a necessidade de se criar um sistema de mapeamento de nomes mais geral e eficiente. Foi neste contexto que surgiu o DNS.

O DNS é (1) um banco de dados distribuído implementado em uma hierarquia de servidores de nome, e (2) um protocolo de camada de aplicação que permite que hospedeiros consultem o banco de dados distribuído. Para tratar a questão da escala, o DNS usa um grande número de servidores organizados, de maneira hierárquica e distribuída. Os principais componentes do DNS são:

- Domain Name Space (Espaço de Nomes): é uma especificação para uma árvore estruturada para armazenar os espaços de nomes. Esta árvore é dividida em ZONAS não-superpostas. Normalmente, cada zona tem um servidor de nome principal (onde os registros são mantidos em disco) e um ou mais servidores secundários;
- Resource Records (Registro de Recursos): A principal função do DNS é mapear nomes de domínios em registros de recursos (não somente endereço IP). Um registro de recurso é uma tupla de cinco campos: Domain_name (normalmente existem muitos registros, de tipos diferentes, para cada domínio); TTL (tempo de vida do registro quando em cache); Class (IN quando relacionado à Internet, outros códigos são raramente encontrados); Type (tipo de registro); e Value (sua semântica depende do tipo de registro). Os principais tipos de registro existentes são apresentados a seguir na Tabela 1.

Tipo	Significado	Valor
SOA	Início de autoridade	Parâmetro para essa zona
A	Endereço IP de um host Inteiro de 32 bits	
MX	Troca de mensagens de correio	Prioridade, domínio disposto a aceitar correio eletrônico
NS	Servidor de nomes	Nome de um servidor para este domínio
CNAME	Nome canônico	Nome de domínio
PTR	Ponteiro	Nome alternativo de um endereço IP
HINFO	Descrição de host	CPU e sistema operacional em ASCII
TXT	Texto	Texto ASCII não-interpretado

Tabela 1: principais tipos de registros de recursos do DNS para o IPv4.

- Name Servers (Servidores de Nomes): são programas servidores que detêm informação sobre a estrutura da árvore de domínio e também tem a capacidade de registrar informações. Resumidamente, há quatro classes de servidores de nomes: servidores de nomes raiz (cerca de 13 servidores); servidores DNS de domínio de nível superior (.int, .com, .mil, .net, .br, .jp, etc. - cerca de 200 servidores); servidores DNS com autoridade (chamados de AUTHORITY - os que detêm os registros de recursos de servidores, localizados em sua zona, que podem ser acessados publicamente); e servidores intermediários;
- Resolvers (Resolvedores): são programas, executados tanto nos clientes quanto nos servidores, que extraem informações dos Name Servers em resposta a requisições feitas por clientes. Esses programas são aptos a responder as consultas diretamente ou então encaminhar a consultas caso a informação não esteja disponível naquele Name Server. São programas que podem ser acessados diretamente por programas de usuários ou rotinas do sistema operacional sem a necessidade de utilização de protocolos;
- Cache DNS: é uma característica muito importante, usada para aumentar o desempenho das respostas as consultas DNS e reduzir o número de mensagens DNS. Em cada elemento da cadeia (cliente, servidores local, secundários, primários, com autoridade, de nível superior etc.) informações (registros) de respostas podem ser armazenadas em memória local. Para evitar o uso de registros desatualizados, é associado a cada registro um número TTL (Time to Live) que determina seu tempo de vida. O TTL deve ser definido de acordo com a frequência de atualização dos registros.

Como foi mencionado na explicação sobre o Cache DNS, sua principal finalidade é aumentar o desempenho das consultas DNS. No entanto, há que se ter cuidado com a desatualização dos registros em cache, de modo que as consultas não seja respondidas erroneamente.

Voltando ao enunciado da questão, foi mencionado *que não há mudanças frequentes de endereços IP e que as consultas, oriundas de usuários legítimos, são relacionadas ao servidor WEB.*

Nessas condições, podemos afirmar que seria possível aumentar TTL das respostas positivas, de modo que as consultas subsequentes sejam respondidas diretamente pelo cache, sem o risco de as informações cache estarem inconsistentes.

Portanto, a resposta da questão é a alternativa B.