Problema 3: Tempo em Sistemas Distribuídos

Calendário

Semana	Data	Grupo Tutorial
1	04/07	Sessão tutorial 1: Apresentação do Problema 3
2	09/07	Sessão de desenvolvimento 1
3	11/07	Sessão tutorial 2
4	16/07	Sessão desenvolvimento 2
5	18/07	Sessão tutorial 3
6	23/07	Apresentação do problema 3

Antes, um pouco de ciência...

Em 1687, o famoso Físico Isaac Newton introduziu em seu tratado “Os Princípios Matemáticos da Filosofia Natural” o conceito de tempo de forma absoluta. O objetivo da discussão foi evidenciar o fato de que o tempo físico é definido em termos de relógios. Para Newton, o tempo não poderia ser afetado por qualquer condição física e não possuía relação com nenhum outro fenômeno externo. Porém, em 1905 Albert Einstein demonstrou de forma teórica, longe de ser percebida em nossa experiência quotidiana, que o tempo é afetado por condições físicas extremas. A sua teoria da relatividade especial foi baseada nas consequências de observações anteriores sobre a velocidade da luz, que é constante para todos os observadores, independentemente de sua velocidade relativa. A partir dessa suposição, ele provou que dois eventos considerados simultâneos em um ponto de referência não são necessariamente simultâneos para observadores em outros pontos de referência, e que estão se movendo em relação a ele. Por exemplo, um observador na Terra e um observador viajando em uma nave espacial, quanto mais suas velocidades relativas aumentassem, discordariam a respeito do intervalo de tempo entre os eventos. A ordem relativa de dois eventos pode ser até invertida para dois observadores diferentes, mas isso não poderia acontecer se um evento tivesse causado a ocorrência do outro. Nesse caso, o efeito físico acompanha a causa física para todos os observadores, embora o tempo decorrido entre causa e efeito possa variar. Assim, a temporização de eventos físicos mostrou-se relativa ao observador, e a noção de Newton de tempo físico absoluto mostrou-se sem fundamento nas condições extremas do universo. Como consequência, não existe no universo nenhum relógio físico especial ao qual possamos recorrer para medir intervalos de tempo. Experimentos modernos, como o transporte de relógios atômicos a velocidades extremamente elevadas e a observação da passagem da luz através de campos gravitacionais extremamente intensos, provaram que tais condições de fato afetam a “marcha” do relógio. A teoria da relatividade, diferente do que a maioria das pessoas acreditam, não é apenas um modelo quântico e matemático que fica no abstratíssimo teórico. Ela foi provada e possui reais impactos no mundo tecnológico.

Agora um pouco de teoria...

De forma análoga ao tempo definido por Newton, um sistema centralizado efetua suas operações baseado em um relógio absoluto, cuja leitura é realizada através de métodos locais por processos para sincronizar operações. Naturalmente, sistemas computacionais e suas aplicações podem ser sensíveis a problemas relativos à sincronização do tempo de diferentes maneiras. Por exemplo, em sistemas orientados a objetos, precisamos estabelecer se as referências para um objeto em particular não existem mais e se o objeto se tornou lixo, no caso em que podemos liberar sua memória. Estabelecer isso exige observações dos estados de processos, para descobrir se eles contêm referências, e dos canais de comunicação entre processos, quando as mensagens contendo referências estejam em trânsito. No caso de um computador isolado, a exatidão de seu relógio em relação à uma referência de tempo pode não ser problemática. Uma importante propriedade a ser garantida é a monotonicidade,

que significa que o tempo sempre avança. Além disso, eventuais ajustes no relógio devem ser, quando possíveis, graduais. Saltos no tempo para o futuro, mesmo de alguns poucos segundos, podem ser ruins, e para o passado, desastrosos. Relógios implementados em software podem ser ajustados, intencionalmente ou não, para um tempo no passado. Por isso, um recurso disponível em sistemas operacionais é a sincronização do relógio local do computador com algum servidor de tempo externo para evitar problemas relacionados ao tempo.

No entanto, medir tempo é problemático nos sistemas distribuídos. Isso não se deve aos efeitos da teoria da relatividade, que são desprezíveis ou inexistentes para computadores convencionais, a não ser que se considere computadores viajando em naves espaciais. Mas de forma análoga a relatividade de Einstein, os relógios de diferentes computadores, assim como os outros relógios, tendem a não estar em perfeito acordo. Os relógios baseados em cristal usados nos computadores estão, assim como todos os outros relógios, sujeitos a derivações (*drift*). Isso significa que eles contam o tempo com diferentes velocidades e, portanto, divergem mesmo que sejam sincronizados em algum momento. Como exemplos de aplicações distribuídas afetadas pelo tempo podemos citar:

- *Sistemas de distribuição de conteúdo*: Usam estampas de tempo para controlar a expiração dos documentos e cache. Servidores com o tempo errado podem causar perda ou impedir o acesso às mesmas;
- *Sistemas de arquivos*: Alguns eventos importantes como a criação e modificação de arquivos são marcados por estampas de tempo. Algumas aplicações dependem dessas informações. Se alguma dessas datas estiver no futuro, as aplicações podem agir de forma indevida, ou mesmo deixar de funcionar por completo. Como exemplos de aplicações sensíveis a essa situação pode-se citar os sistemas de controle de versão, sistemas de compilação, sistemas de backup e de banco de dados;
- *Agendamento de eventos*: Aplicações como o *crontab* dependem do tempo correto para funcionarem;
- *Criptografia*: Muitas técnicas criptográficas fazem uso de estampas de tempo para os eventos e chaves para prevenir alguns tipos de ataques. Se os computadores envolvidos não estiverem sincronizados entre si, a autenticação e comunicação criptografada podem falhar;
- *Protocolos de comunicação e aplicações de tempo real*: essas aplicações, que incluem as interfaces gráficas, fazem uso de filas de eventos, *timeouts*, *timers*, e outros recursos de software ligados ao tempo.
- *Sistemas transacionais e bancos de dados distribuídos*: Dependem de relógios exatos e muitas vezes, de sua sincronia com a hora legal. Por exemplo, bolsas de valores tem horários bem definidos de início e término do pregão e que devem ser respeitados. A Receita Federal aceita as declarações de Imposto de Renda geralmente até a meia noite da data limite para a entrega.

E finalmente o problema...

A tarefa é desenvolver uma solução distribuída para a sincronização aproximada de um grupo definido de relógios físicos. Um dos relógios deve ser tomado como referência para todos os outros. Caso esse relógio venha a falhar ou apresentar problemas, um novo relógio, dentre todos os existentes no grupo, deve ser escolhido como o novo servidor de tempo de forma a manter a monotonicidade do sistema levando em consideração as derivações de todos os relógios. Os relógios podem estar em qualquer lugar da Internet ou do Universo. Para facilitar a implementação, os efeitos da teoria da relatividade de Einstein não precisam ser considerados.

Observação:

Para facilitar o desenvolvimento e os testes da solução no laboratório de redes, o tempo não precisa ser obtido através de uma função do sistema operacional. Um relógio virtual deve ser implementado através de um contador incrementado segundo a segundo. Na interface, o tempo pode ser alterado e um valor virtual de *drift* pode ser inserido para cada relógio. Por exemplo, em vez de ser atualizado a cada segundo, o contador (relógio) pode ser incrementado a cada 1,1 ou 0,9 segundos. Naturalmente, o *drift* real do relógio de software vai incluir um intervalo aleatório de tempo em microssegundos relativo a outros tipos de processamentos como, por exemplo, o tempo relativo ao escalonamento do sistema operacional sobre o processo do relógio.

Nossas Regras

- Os alunos devem se reunir em dupla para realizar o trabalho, com possibilidade de formar apenas 1 trio quando não for possível formar uma dupla.
- O funcionamento do sistema deverá ser apresentado no laboratório de Redes e Sistemas Distribuídos ou em outro local a combinar. Durante a apresentação, o tutor realizará arguição aos membros da equipe.
- O relatório pode ser entregue no github, devidamente referenciado.
- O código fonte deve ser entregue devidamente comentado.
- Uso de docker, como sempre.
- O **prazo final** de entrega do trabalho será dia .

Observações:

- **Trabalhos entregues fora do prazo serão penalizados com 20% do valor da nota + 5% por dia de atraso**. Esse atraso deve ser na mesma semana da entrega final.
- Trabalhos copiados da INTERNET ou de qualquer outra fonte e trabalhos iguais terão nota ZERO.
- As informações para solução do problema podem ser ESCLARECIDAS ou ALTERADAS no decorrer das sessões.
- O ambiente de demonstração da solução do problema deve ser instalado previamente no Laboratório de Redes e Sistemas Distribuídos, de modo a otimizar o tempo das apresentações.

Avaliação

A nota final será a composição de 34 notas.

1. Desempenho individual (20%)
2. Relatório (30%)
3. Produto Final (código incluso) (50%)

Obs.: A nota do produto será única devendo os participantes do grupo indicar ao tutor o percentual da nota para cada um dos membros.

Referências:

COMER, D. E. **Redes de Computadores e Internet**. 4. ed. São Paulo: Bookman, 2007.

COULOURIS, F. G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas Distribuídos: Conceitos e Projeto**. 4. ed. Bookman, 2007.

KUROSE, J.; ROSS, K.; **Redes de Computadores e a Internet: uma Abordagem Top-Down**. 5 ed. São Paulo: Pearson, 2010.

DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**. 6 ed. São Paulo: Bookman, 2005. 1152 p.

COSTA, D. G. **Java em Rede: Recursos Avançados de Programação**. Rio de Janeiro: Brasport, 2008.

TANENBAUM, A. S.; STEEN, M. V. **Sistemas Distribuídos: Princípios e Paradigmas**. 2. ed. São Paulo: Person-Prentice Hall, 2007.