SistemasDistribuidos-Lista3

1.

- Processo que chama e processo que executa função estão em máquinas diferentes, acarretando em espaços de endereçamento e ambiente computacional distintos;
- Representação de dados diferentes (Little Endian, Big Endian);
- Falhas de rede e das máquinas envolvidas.

Extra: Little Endian vs. Big Endian

Sistemas representam vetores de bytes de maneiras diferentes. Considerando que temos uma *word* (32 *bits*) representada em hexadecimal por **0A0B0C0D**:

- Big Endian (**MSB primeiro**): Vai representar na memória (m[]) como m[0] = 0A; m[1] = 0B; m[2] = 0C; m[3] = 0D;
- Little Endian (**LSB primeiro**): Vai representar na memória (m[]) como m[0] = 0D; m[1] = 0C; m[2] = 0B; m[3] = 0A;

2.

```
1. Cliente faz chamada de função para servidor de RPC e espera (bloqueante) mensagem de aceitação;
2. Servidor de RPC recebe chamada de função e envia mensagem de aceitação para cliente. Servidor passa a
3. Cliente recebe aceitação, a chamada de função retorna mas ainda não há resultado. Cliente continua su
4. Ao terminar o procedimento, servidor envia resultado ao cliente via RPC;
5. Mensagem de resultado gera uma interrupção no cliente, que por sua vez envia o reconhecimento de que
```

Extra: *Marshalling*, *Unmarshalling* e *Stubs*

Para evitar problemas por divergências nas representações de dados entre os participantes de um sistema de RPC, é feito o *marshalling* de mensagens a serem enviadas, convertendo a mensagem para uma representção de comum acordo entre os participantes. Ao receber uma mensagem, é feito o *unmarshalling* da mesma.

Tanto o *marshalling* quanto o *unmarshalling* são feitos pelos *stubs*, que são trechos de código fornecidos pelas biblioecas de RPC e tornam o envio e recebimento de chamadas e retornos de RPC transparentes aos usuários.

3.

A sincronização de relógio resolve o problema da *odenação global de eventos* para as máquinas cujos relógios foram sincronizados. Um exemplo de situação gerada por esse problema de ordenação em Sistemas Distribuídos é:

- Dois usuários compartilam um diretório no Dropbox;
- O **usuário 1** envia a versão **v1** de um documento e seu computador marca *12:00* no momento do envio;
- O **usuário 2** envia a versão **v2** do mesmo documento *5 minutos* depois, mas o relógio do computador dele está *1 hora* atrasado em marca *11:05* na hora do envio;
- A versão mais recente para o Dropbox fica como **v1**.

Caso os relógios dos usuários estivessem sincronizados no exemplo anterior, o problema não ocorreria.

4.

A *sincronização de relógios* tem **dois problemas principais**: Qual é a hora certa e como fazer a sincronização.

O problema da hora certa é resolvido utilizando padrões internacionais e protocolos para definir uma hora certa para todos, sendo UTC (Universal Time Coordinated) o padrão utilizado atualmente. O mesmo é baseado em dois componentes: o TAI (International Atomic Time), e combina os tempos de cerca de 400 relógios atômicos de alta precisão espalhados pelo mundo, e o UT1 que é referente à rotação da Terra.

Já o método de sincronização atual para computadores é o Network Time Protocol (NTC) que utiliza UTC como referência. O sistema funciona como uma hierarquia de servidores que vão da maior pra menor precisão. Nesse sistema, clientes solicitam a hora periodicamente a três ou mais servidores e determina sua hora utilizando uma média com filtros (para a remoção de *outliers*) dos valores obtidos. A hora não é alterada imediatamente, para evitar saltos temporais e a volta no tempo. O que ocorre é que ataxa de progressão do relógio é ajustada para que o relógio local se ajuste suavemente para UTC.

5.

Considerando dois computadores A e B, B quer usar o relógio de A como referência para a sincronização:

1. B envia uma solicitação em $T1$ (tempo registrado em B) para A, guardadndo o valor de $T1$;
2. A recebe a solicitação em $T2$, processa a mesma e envia a resposta em $T3$ para B, mandando na mensagem os valores de $T2$ e $T3$ (tempos registrados em A);
3. B recebe a resposta em $T4$ (tempo registrado em B);
4. Assumindo que o retardo de ida é igual ao de volta, B calcula o mesmo como $d = ((T_4 - T_1) - (T_3 - T_2))/2$ e ajusta seu relógio para $T_3 + d$.

Dessa maneira, B é sincronizado a A.

6.

O Network Time Prtocol (NTP) funciona para sincronizar relógios com o padrão UTC e é estrutuado com uma hierarquia de servidores e clientes. No topo da hierarquia estão os computadores com relógios de maior precisão. Descendo na hierarquia, a precisão diminui.

Um determinado computador em um nível da hierarquia solicita a hora periódicamente para três ou mais servidores de uma hierarquia superior. Ele faz então uma filtragem dos resultados, para remover *outliers*, tira uma média e ajusta sua taxa de progressão para se ajustar suavemente ao horário padrão.

7.

1. Relações $x \to y$:
   - $a \to r$
   - $h \to b$
   - $i \to n$
   - $s \to j$
   - $c \to o$
   - $d \to t$
   - $k \to e$
   - $p \to f$
   - $u \to l$
   - $g \to q$
2. Uma relação $x \parallel y$:

- $p \parallel u$

3. Relações no conjunto $E = b, k, n, u$:

   - $b \parallel k$
   - $b \parallel n$
   - $b \rightarrow u$: $b \rightarrow c$, $c \rightarrow d$, $d \rightarrow t$, $t \rightarrow u$
   - $k \parallel n$
   - $k \parallel u$
   - $n \parallel u$

4. Relógio para cada evento:

   - L(a) = 1
   - L(b) = 2
   - L(c) = 3
   - L(d) = 4
   - L(e) = 6
   - L(f) = 10
   - L(g) = 11
   - L(h) = 1
   - L(i) = 3
   - L(j) = 4
   - L(k) = 5
   - L(l) = 8
   - L(m) = 1
   - L(n) = 4
   - L(o) = 8
   - L(p) = 9
   - L(q) = 12
   - L(r) = 2
   - L(s) = 3
   - L(t) = 5
   - L(u) = 7

5.

8.

9.

10.

11.

12.

13.

14.

15.

16.

17.

18.

19.

20.

21.

22.

23.

Revisar:

- aula 11
  - Falhas e Semântica em RPC;
  - RMI: vantagens e desvantagens
-