Engenharia Informática

Paradigmas de Sistemas Distribuídos

Exame[1]

20 de junho de 2022 Duração: 1h15m

I

1 Discuta a necessidade de usar vários *threads* num programa por eventos para um servidor em grande escala.

2 Considere um sistema de recolha e processamento de mensagens de diagnóstico (*logs*) de um sistema distribuído. Admite-se que cada um dos servidores monitorizados se liga ao sistema de recolha e emite mensagens constituídas por uma linha de texto. O sistema de recolha tem então que: eliminar duplicados sucessivos vindos da mesma origem; verificar se a mensagem deve desencadear um alarme; adicionar hora e data; comprimir a mensagem; e armazená-la em disco. Ordene de acordo com a conveniência para o programador na resolução deste problema, justificando sucintamente, os mecanismos de programação por eventos que estudou.

II

1 A programação clássica (bloqueante, não *event-driven*) é mais fácil/intuitiva para os programadores, mas bastante limitada em termos de interagir com informação proveniente de várias fontes. Explique como o modelo de atores tal como instanciado em Erlang permite obter "o melhor de dois mundos", em termos de flexibilidade e simplicidade da programação.

2 Explique como usaria o `receive` em Erlang, de modo detetar se um processo "servidor", em ciclo, já não recebe mensagens há mais de um minuto, atualizando um parâmetro `Inactive` de modo apropriado. Tente evitar que o processo corra sem necessidade.

```
loop(Active, OtherState) ->
  receive
    ...
  end.
```

3 Considere que pretende repetidamente, e potencialmente com grande frequência, enviar mensagens, cada uma para vários nós, para cada um destes processar cada mensagem. Comente sobre a adequação de usar sockets ZeroMQ do tipo PUB e SUB neste contexto. Diga, justificando, que combinação de sockets ZeroMQ escolheria.

[1] Cotação — 20