embarcadero

Embarcadero
Delphi Conference

Controlando a Concorrência em Aplicações Multi-Thread

Mário Guedes

O QUE É THREAD?

Embarcadero
Delphi Conference

Definições

- Thread, ou segmento, é a unidade básica de execução de um Sistema Operacional moderno;
- Um software tem, no mínimo, uma thread: *A thread principal*;
- Para que um software possa fazer mais de uma tarefa ao “mesmo tempo” devemos criar nossas próprias threads;

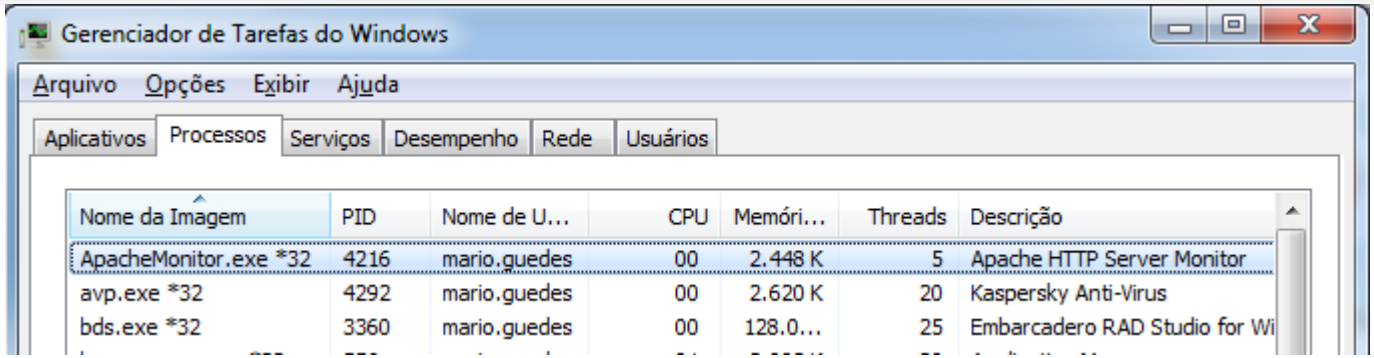
“Ao mesmo tempo” entre aspas!

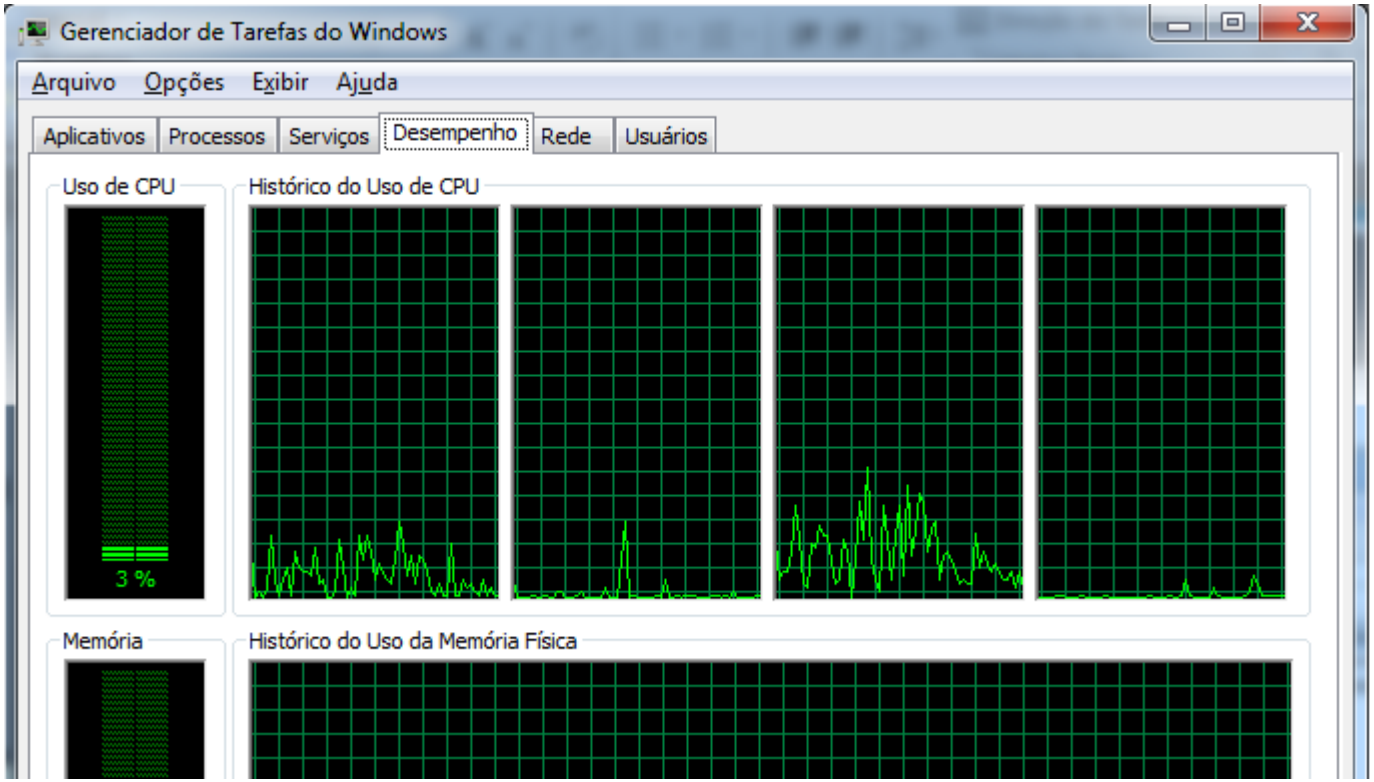
- Devemos ter consciência de que o Sistema Operacional gerencia as execuções das threads e que só uma thread é executada a um só tempo;

- Só quando temos mais de um núcleo é que o Sistema Operacional estará executando várias threads, efetivamente, ao mesmo tempo:
  4 núcleos = 4 threads simultâneas

Gerenciador de tarefas do Windows

Embarcadero Delphi Conference





Vantagens em utilizar Threads

- Hoje os equipamentos tem vários núcleos e nossos softwares podem, efetivamente, tirar proveito disso;
  - Velocidade de execução;
  - Escalabilidade;
  - Tarefas em segundo plano;
  - Monitoração de eventos;
  - etc;

Classe TThread

- No Delphi temos a classe **TThread** presente na unit ***System.Classes***;
- Ela abstrai as chamadas à API do Windows para se criar uma thread;
- Possui propriedades e métodos para o efetivo controle da thread;
- Basicamente devemos reescrever o método **Execute** mantendo a thread em *loop* se for o caso (propriedade **Terminated**);
- E de uma vez por todas: **TTimer não é uma thread**.

Máscara de afinidade

Para que duas ou mais threads sejam executadas efetivamente ao mesmo tempo e portanto tenha o desempenho melhorado, podemos definir a afinidade da thread com um ou mais núcleos:

```
cRet := SetThreadAffinityMask(GetCurrentThread, 1); //Núcleo 1
 if (cRet = 0) then
 begin
   cRet := GetLastError;
   sErro := SysErrorMessage(cRet);
 end;
```

http://msdn.microsoft.com/en-us/library/windows/desktop/ms686247(v=vs.85).aspx

PROBLEMÁTICA DA CONCORRÊNCIA

Embarcadero
Delphi Conference

Recursos críticos

- Quando optamos por trabalhar com threads temos que gerenciar os recursos críticos;

- Por recurso crítico entenda tudo aquilo que só pode ser manipulado por apenas uma thread a um só tempo, como por exemplo:
  - Componentes da VCL;
  - Tabelas;
  - Arquivos (em especial escrita);
  - Conexão TCP/IP;
  - etc;

Contexto de execução

- Ao se trabalhar com alguns *frameworks* como por exemplo o Data Snap, podemos estar dentro de uma thread, apesar de mão termos usado explicitamente a classe TThread;

- Isso evidencia a necessidade de tomarmos um maior cuidado com recursos críticos, também referenciado por recursos *compartilhados*.

Componentes da VCL

- A classe TThread possui o método ***Synchronize*** que sincroniza o processamento com a thread principal;
- Neste cenário, é a thread principal que executa o comando sincronizado;
- Os componentes da VCL essencialmente manipulam as mensagens do Windows;
- Se uma thread precisa manipular elementos da interface essa manipulação deve ser feita via *Synchronize*.
- Deve-se evitar que uma thread faça usa massivo de elementos de interface pois perde-se o ganho que a thread oferece;
- O uso do *Synchronize* degrada a performance do sistema, portanto use apenas *quando* e *onde* for necessário;

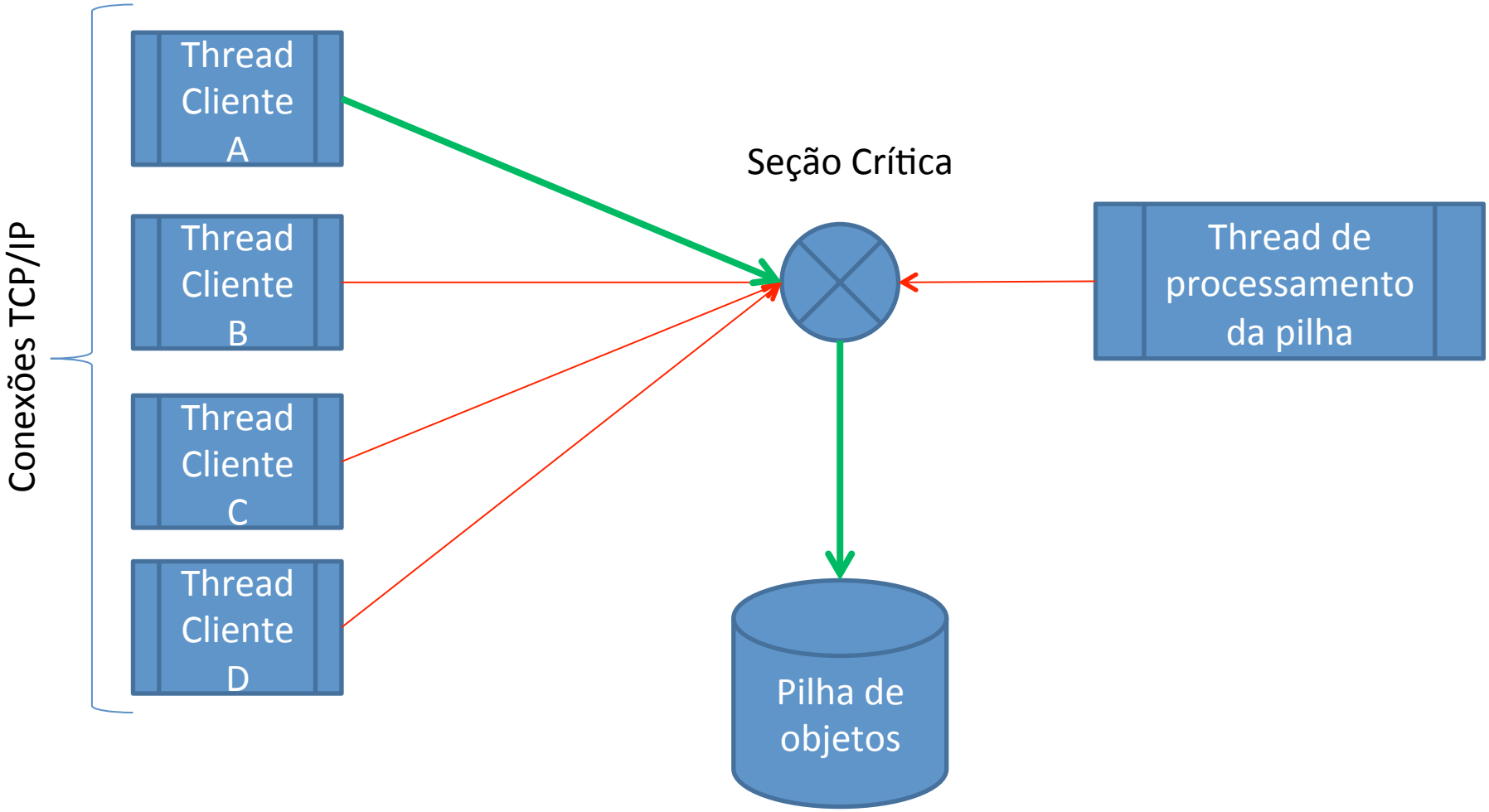
Exemplo prático

Exemplo prático #1

Seção crítica

- Há elementos que não devem ser acessados ao mesmo tempo por mais de uma thread mas que não envolve elementos da VCL e portanto não precisam ser sincronizados;
- Para isto podemos utilizar a classe **TCriticalSection** presente na unidade ***System.SyncObjs****;*
- Funciona como um semáforo: *abre passagem para um, fecha passagem para outros;*
- Para entrar na seção crítica usamos o método **Enter** . Se a seção já estiver ocupada ficaremos aguardando até que a mesma seja liberada pela outra thread;
- Para liberar a seção crítica usamos o método **Release** ;

Seção crítica: cenário de exemplo



Exemplo prático

Embarcadero
Delphi Conference

Exemplo prático #2

Sinalização para Thread

- Um fator importante no desenvolvimento multi-thread é quando a thread fica ociosa aguardando algum evento;
- Em geral ficamos em um loop infinito o que consome recursos do ambiente;
- Podemos então usar a classe **TEvent** , que nos permite aguardar uma sinalização externa;
- A thread fica hibernando até ser acionado pelo evento, o que é feito a partir do método **SetEvent**;
- Importante frisar que o nome passado na criação do TEvent deve ser único em todo o Sistema Operacional!

Exemplo prático

Exemplo prático #3

Incremento de variável do tipo Integer

- Eventualmente o recurso crítico é a geração de um ID por exemplo, resultado do incremento de uma variável integer;
- Se duas threads executarem o método ao mesmo tempo poderão ler o mesmo valor e escrevendo o mesmo incremento;
- Neste caso específico podemos usar os métodos **InterlockedIncrement** , **InterlockedDecrement** além de outras opções;
- É uma *function* que faz a operação desejada e retorna o valor;

Aguardando várias threads

- Ao se dividir um processamento em thread, pode ser necessário aguardar que todas finalize para dai iniciar um processo posterior;
- A instrução **WaitForMultipleObjects** é uma ótima alternativa para isto;
- Basicamente passa-se para a função um array de Handles das threads envolvidas;
- Quando um objeto TThread finaliza é efetuado um **CloseHandle** e com isso o **WaitForMultipleObjects** é sinalizado.

Exemplo prático

Exemplo prático #4

TMonitor

- Simplificando vários conceitos, a unidade **System** possui o record **TMonitor** que possui métodos bastante interessantes, entre eles:
  - **Enter:** *Entra em uma seção crítica usando um objeto como parâmetro;*
  - **Exit:** *Sai da seção crítica, usando um objeto como parâmetro;*
  - **Wait:** *Aguarda a liberação da seção crítica;*
  - **Pulse:** *Notifica a próxima thread que está aguardando a liberação da seção crítica;*

Exemplo prático

Exemplo prático #5

Existem outras alternativas

- Sim, o que foi mostrado aqui são apenas algumas opções – que atende à maioria das necessidades ;
- Existem outras com características diferentes que se adequam melhor a uma ou outra necessidade;
- Um outro tema correlato é a comunicação entre processos;

Links interessantes

- Balaio Tecnológico
  Luís Gustavo Fabbro
  http://balaiotecnologico.blogspot.com.br/2009/08/trabalhando-com-threads.html

- About.com Delphi Programming
  Zarko Gajic
  http://delphi.about.com/od/kbthread/Threading_in_Delphi.htm

- Eu Gosto do Delphi
  Mário Guedes
  http://eugostododelphi.blogspot.com.br/search/label/thread

Perguntas?

Embarcadero Delphi Conference

- Portal de Treinamentos e Vagas – http://www.edobrasil.net/treinamentos
- Embarcadero Developer Network - http://edn.embarcadero.com
- Diretório de MVP's - http://www.embarcadero.com.br/mvp-directory
- Documentação dos Produtos - http://docs.embarcadero.com
- CodeRage 7 - http://www.embarcadero.com/coderage
- YouTube - http://youtube.com/user/embarcaderodobrasil
- Twitter - https://twitter.com/EmbarcaderoBR
  http://twitter.com/EmbarcaderoTech
- Blogs: http://blogs.embarcadero.com
- Facebook:
  https://www.facebook.com/pages/Embarcadero-Delphi-Brasil/399151510134179

- atendimento@embarcadero.com.br
- (11) 5643-1333

Obrigado!

Embarcadero Delphi Conference

Mário Guedes

mario.guedes@arrayof.com.br

http://eugostododelphi.blogspot.com
http://br.linkedin.com/in/jmarioguedes
http://facebook.com/eugostododelphi
http://twitter.com/eugostododelphi