**BENEFIX**

2011

**Relatório de Atividades**

**Projeto Z10**

[Sumário de Atividades 3](#_Toc322008093)

[Atividade executadas no MainFrame 3](#_Toc322008094)

[Atividade executadas no PC/Linux 3](#_Toc322008095)

[Atividade executadas no PC Windows 4](#_Toc322008096)

[Atividade executadas teste da Aplicação 4](#_Toc322008097)

[Ambiente de Testes 5](#_Toc322008098)

[Especificações das Máquinas 5](#_Toc322008099)

[Considerações 5](#_Toc322008100)

[Configuração Mono – PC 6](#_Toc322008101)

[Objetivo 6](#_Toc322008102)

[Instalação LinuxPC 6](#_Toc322008103)

[Donwload Suse Linux Enterprise 6](#_Toc322008104)

[Instalação Pacotes Pré Compilados 6](#_Toc322008105)

[Instalaçãodo Mono apartir dos fontes 8](#_Toc322008106)

[Download do Fonte do Mono 8](#_Toc322008107)

[Compilação do Mono 9](#_Toc322008108)

[Instalação do Apache 10](#_Toc322008109)

[Configuração Mono – Z10 11](#_Toc322008110)

[Objetivo 11](#_Toc322008111)

[Preparação do Ambiente 11](#_Toc322008112)

[Testes iniciais com Z10 12](#_Toc322008113)

[Bogomips 12](#_Toc322008114)

[Calculo de PI 13](#_Toc322008115)

[Compactação de Arquivos 14](#_Toc322008116)

[Teste Ruby Destrossator 15](#_Toc322008117)

[Instalação Linux Z10 15](#_Toc322008118)

[Instalando Z10 15](#_Toc322008119)

[Instalando Pacotes Mono Extension 15](#_Toc322008120)

[Atualização do Mono Apartir dos Fontes 16](#_Toc322008121)

[Instalação e Configuração do Apache 18](#_Toc322008122)

[Compatibilidade do WebISS 19](#_Toc322008123)

[Compatibilidade de Código 19](#_Toc322008124)

[Compatibilidade de Bibliotecas 19](#_Toc322008125)

[Compilando o WebISS no Linux 20](#_Toc322008126)

[Formulários com Erros 20](#_Toc322008127)

[Formulários que contem MagicAjax 21](#_Toc322008128)

[Compatibilidade do código Fonte WebIss 21](#_Toc322008129)

[Problema de Execução na camada de Persistência 22](#_Toc322008130)

[Componentes de Relatórios 24](#_Toc322008131)

[Considerações Sobre Compatibilidade 27](#_Toc322008132)

[Instalando o WebISS 28](#_Toc322008133)

[Deploy do WEBIIS no Linux 28](#_Toc322008134)

[Deploy do WEBIIS no LinuxZ10 28](#_Toc322008135)

[Deploy dos Web Services - DesWebService 29](#_Toc322008136)

[Testes do WebISS 30](#_Toc322008137)

[Planejamento dos testes 30](#_Toc322008138)

[Testes unitários 30](#_Toc322008139)

[Construção Teste unitários de desempenho 31](#_Toc322008140)

[Teste de acesso a banco SQL 31](#_Toc322008141)

[Teste de Algoritmos Genéricos 33](#_Toc322008142)

[Algoritmo multi thread 34](#_Toc322008143)

[Criação de Objetos em Memória 35](#_Toc322008144)

[Teste SQL – 1 Thread .C# 36](#_Toc322008145)

[Testes dos WCF – Nota Eletrônica 37](#_Toc322008146)

[Resultados Encontrados 38](#_Toc322008147)

[Como Garantir Qualidade 39](#_Toc322008148)

[Suporte em mainframe 39](#_Toc322008149)

[Código Fontes de Teste 40](#_Toc322008150)

[Código fonte do Programa de Cálculo de PI 40](#_Toc322008151)

[Código Fonte Ruby Destrossator 43](#_Toc322008152)

[Códigos de Algoritimos Genéricos C# 46](#_Toc322008153)

[Código Fonte Acesso Banco de Dados C# - Single Thread 49](#_Toc322008154)

[Código Fonte Java de Escalabilidade 56](#_Toc322008155)

[Slides da Apresentação 75](#_Toc322008156)

Sumário de Atividades

Atividade executadas no MainFrame

|  |  |
| --- | --- |
| **Ambiente Mainframe** | Status |
| Instalação das Ferramentas básicas para configuração e compilação do Mono | OK |
| Configuração do Ambiente Web – Apache | OK |
| Instalação de Pacotes Prontos Mono | OK |
| Download do código fonte do Mono | OK |
| Compilação de toda plataforma Mono | OK |
| Execução do sistema de teste e realizar medição de tempos de execução para comparação | OK |
| Configuração do modo de execução do mono para a ambiente WEB: MODMONO, | OK |
| Configuração do modo de execução do mono para o ambiente WEB: FastCGI | OK |
| Preparação do ambiente para executar a aplicação WEBISS | OK |
| Criar alguns scripts de teste simples para medição inicial de desempenho estanto IO, Processamento, Memória. | OK |
| Construir scripts de compilação e instalação automatizados para facilitar e documentar o trabalho. | OK |
| Instalar WebIss e iniciar testes | OK |
| Gera um documento de gestão de configuração para documentar a instalação e problemas encontrados. | OK |

Atividade executadas no PC/Linux

|  |  |
| --- | --- |
| **Ambiente PC AMD** | Status |
| Donwload do Suse 11 | OK |
| Instalação Suse 11 em VMs de Teste | OK |
| Instalação Suse 11 em Sevidor Local | OK |
| Comparação de Velocidade Maquinas Locais com o MainFrame para verificar se estão compatíveis | OK |
| Instalação das Ferramentas básicas para configuração e compilação do Mono | OK |
| Configuração do Ambiente Web – Apache | OK |
| Instalação de Pacotes Prontos Mono | OK |
| Download do código fonte do Mono | OK |
| Compilação de toda plataforma Mono | OK |
| Execução do sistema de teste e realizar medição de tempos de execução para comparação. | OK |
| Configuração do modo de execução do mono para o ambiente WEB: ModMono | OK |
| Configuração do modo de execução do mono para o ambiente WEB: FastCGI | OK |
| Preparação do ambiente para executar a aplicação webIss | OK |
| Criar alguns scripts de teste simples para medição inicial de performance estanto IO, Processamento, Memória | OK |
| Construir scripts de compilação e instalação automatizados para facilitar e documentar o trabalho. | OK |
| Instalar WebIss e iniciar testes. | OK |
| Gera um documento de gestão de configuração para documentar a instalação e problemas encontrados. | OK |

Atividade executadas no PC Windows

|  |  |
| --- | --- |
| **Ambiente PC Windows** | Status |
| Instalar o Windows Server | OK |
| Configuração do Ambiente Web – IIS | OK |
| Execução do sistema de teste e realizar medição de tempos de execução para comparação | OK |
| Configuração do modo de execução do .NET para o ambiente WEB | OK |
| Preparação do ambiente para executar a aplicação webIss | OK |
| Instalar WebISS e iniciar testes. | OK |
| Gera um documento de gestão de configuração para documentar a instalação e problemas encontrados. | OK |

Atividade executadas teste da Aplicação

|  |  |
| --- | --- |
| **Testes Da aplicação** | Status |
| Execução de teste de compatibilidade Mono | OK |
| Definição da arquitetura de teste | OK |
| Construção Teste unitários de desempenho com saídas dos tempos medidos | OK |
| Criar teste de SQL Server | OK |
| Teste de Compatibilidade e Funcionamento do Sistema | OK |
| Criar teste de IO | OK |
| Criar teste de Processamento | OK |
| Executar os testes no Mainframe | OK |
| Executar os testes no PC Linux | OK |
| Executar os testes no PC Windows | OK |
| Diagnóstico de problemas mono e descrição de possíveis soluções | OK |
| Teste de compatibilidade do WFC na aplicação WebISS | OK |

Ambiente de Testes

Especificações das Máquinas

Foi estruturado um ambiente de teste com as seguintes configurações de servidores para os testes do WebISS na plataforma Mono. Sendo um sevidor com Suse Linux e Windows Server instalado no laboratório da G6. E uma máquina virtual no Z10 rodando a mesma versão dos Suse Linux.

Considerações

A máquina AMD em questão é uma máquina de cinco anos atrás, um desktop. Considerando uma tecnologia ultrapassada frente a tecnologias de máquinas atuais como processadores XEON e I7 para servidores que atualmente rodando a aplicação do WebISS .

Configuração Mono – PC

Objetivo

O objetivo de realizar a instalação do Mono em plataforma PC é realizar teste do mono no ambiente Linux e no ambiente Windows. Rodando o WebISS e

Instalação Linux PC

A atividade iniciou instalando e configurando um Suse Linux 11 em uma máquina AMD comum, a estratégia para o teste do WebISS no Mono consistem em instalar o Mono num ambiente conhecido, atualizar e compilar todos os pacotes usando a última versão do mono. Feito isso, configurar todo o ambiente do Apache para a execução do mono como o WebIss, fazer o deploy do WebIss e resolver todos os problemas de compatibilidades encontrados. Feito isso reproduzir a mesma instalação e configuração no Mainframe.

# Donwload Suse Linux Enterprise

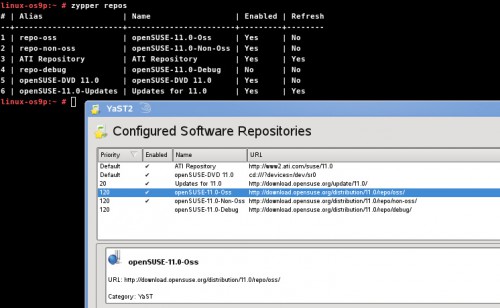
O Suse Linux Enterprise pode ser conseguido numa versão de avaliação no site WWW.suse.com. Podem ser encontradas as versões i586 e 64BITS. Além do Suse, é importante fazer o Donwload do Mono Tools Extension que é uma extensão do Suse que já traz o Mono pré-empacotado. Sem ele não será possível instalar os pacotes do mono para o suse. Outra opção que pode ser usada é a instalação do mono através dos repositórios do mono na internet, como pode ser visto a seguir.

# Instalação Pacotes Pré Compilados

O Linux tem um sistema de empacotamento que é possível usar para instalar os aplicativos. Pacotes pré-compilados pode ser instalados destes repositórios. O comando responsável por instalar pacotes e adicionar repositórios é o Zypper. O uso do Zypper é na verdade bem similar ao do apt-get, com parâmetros e opções similares. Se quiser ver detalhes sobre o pacote antes de instalar, use o "info", como em:

# zypper repos

Os repositórios que aparecem na lista são os mesmos que aparecem ao acessar o "Software > Software Repositories" do YaST.



Você pode tanto adicionar novos repositórios através do YaST quanto adicioná-los diretamente via linha de comando. Nesse caso, usamos o comando "zypper addrepo", Para adicionar o repositório do mono:

# zypper addrepo <http://download.opensuse.org/repositories/Mono/SLE_11/i586/> mono

Lista todos os repositórios já definidos.

# zypper lr

Para instalar um pacote, use o "zypper install", para instalar o mono segue a seguite linha:

#zypper in mono-complete

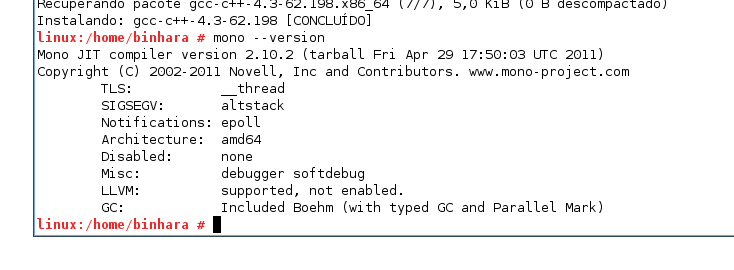
Como de praxe, ele calcula as dependências e exibe uma lista de outros pacotes que precisarão ser instalados, juntamente com o volume de arquivos que serão baixados e o espaço em disco que será usado. Para remover, use o "zypper remove", como em:

# zypper remove mono-complete

Em casos onde você não saiba o nome exato do pacote, ou ele apareça com um nome diferente do que você esperava, use o parâmetro "search" informando um trecho do nome para realizar uma busca, como em:

# zypper search yast

Para verificar se o mono está realmente instalado e em funcionamento bastar abrir um terminar e digitar #mono –version. Deverá ser apresentada a seguinte tela :



Instalaçãodo Mono apartir dos fontes

Outra forma de instalar o Mono é apartir dos códigos fontes. Neste caso é necessários um processo de compilação do fonte. Esse processo é usando geralmente para desenvolvedores que precisa usar a versão em desenvolvimento do mono. Para compilar o Mono apartir dos fontes precisamos instalar os pacotes prontos ou fazer a instalação do bootstrap que é um conjunto mínimo dos executáveis necessários para a compilação do mono.

# Download do Fonte do Mono

O fonte pode ser óbito na forma de pacotes tag.gz site do projeto em <http://download.mono-project.com/sources/> ou a outra forma é baixando diretamente dos repositórios no gitHub em <https://github.com/mono/mono> basta clicar em download e escolher o formato que se deseja baixar .tar.gz ou .zip . Para descompacta o pacote tar.gz usa-se :

# tar –zxvf nome.do.arquivo.tar.gz

Para realizar teste com a versão em desenvolvimento do momento. É melhor usar o git, ele deve ser instalado nos suse com adicionando o seguinte repositório a ser adicionado ao yast.

# zypper addrepo <http://download.opensuse.org/repositories/devel:/tools:/scm/SLE_11_SP1/> Git

No caso do z10 talvez seja necessário compilar o GIT. Para evitar mais uma compilação é mais facil baixar o fonte na máquina local e enviar para o Z10. O comando para baixar o fonte do mono do GitHub:

# git clone https://github.com/mono/mono.git

Após essa operação cerca de 315MB de código fonte serão baixados, e poderá ser dando inicio ao processo de compilação.

# Compilação do Mono

Para compilar o mono você ainda precisar instalar o automake e g++ com o comando. Para essa instalação será necessário que a mídia de DVD do Suse esteja montada.

# zypper in automake

# zypper install gcc-c++

Entre no diretório do mono e inicie a compilação com o comando :

cd mono

./autogen.sh --prefix=/usr/local

make

make install

Agora com a compilação do mono realizada será possível instalar os módulos do mono no Apache e realizar a configuração do ambiente Apache. Em alguns casos será necessários recompilar o apache e seus módulos incluindo o modMono. Destaca-se as configurações do modMono que ficam no /etc/apache2/conf.d/mod\_mono.conf

# mod\_mono.conf

# Achtung! This file may be overwritten

# Use 'include mod\_mono.conf' from other configuration file

# to load mod\_mono module.

<IfModule !mod\_mono.c>

LoadModule mono\_module /usr/lib64/apache2/mod\_mono.so

</IfModule>

<IfModule mod\_headers.c>

Header set X-Powered-By "Mono"

</IfModule>

AddType application/x-asp-net .aspx

AddType application/x-asp-net .asmx

AddType application/x-asp-net .ashx

AddType application/x-asp-net .asax

AddType application/x-asp-net .ascx

AddType application/x-asp-net .soap

AddType application/x-asp-net .rem

AddType application/x-asp-net .axd

AddType application/x-asp-net .cs

AddType application/x-asp-net .vb

AddType application/x-asp-net .master

AddType application/x-asp-net .sitemap

AddType application/x-asp-net .resources

AddType application/x-asp-net .skin

AddType application/x-asp-net .browser

AddType application/x-asp-net .webinfo

AddType application/x-asp-net .resx

AddType application/x-asp-net .licx

AddType application/x-asp-net .csproj

AddType application/x-asp-net .vbproj

AddType application/x-asp-net .config

AddType application/x-asp-net .Config

AddType application/x-asp-net .dll

DirectoryIndex index.aspx

DirectoryIndex Default.aspx

DirectoryIndex default.aspx

MonoAutoApplication enabled

#MonoServerPath /usr/bin/mod-mono-server2

MonoApplications "/:/srv/www/htdocs"

O particionamento desta máquina ficou distribuído de uma forma muito simples:

/dev/disk/by-id/ata-SAMSUNG\_HD250HJ\_S0URJ9BPB13601-part3 swap swap defaults 0 0

/dev/disk/by-id/ata-SAMSUNG\_HD250HJ\_S0URJ9BPB13601-part2 / ext3 acl,user\_xattr 1 1

proc /proc proc defaults 0 0

sysfs /sys sysfs noauto 0 0

debugfs /sys/kernel/debug debugfs noauto 0 0

usbfs /proc/bus/usb usbfs noauto 0 0

devpts /dev/pts devpts mode=0620,gid=5 0 0

# Instalação do Apache

Para instalar o apache, mas usar a ferramenta de instalalação padrão do Suse o Yast e selecionar os pacotes do apache. No suse ja vem com os módulos do mono pré compilados para o apache. Basta seleciona-los e realizar a instalação.

Caso se use uma instalacao do mono apartir dos fontes, será necessário recompilar os módulos juntamente com o Apache.

Configuração Mono – Z10

Objetivo

O objetivo desta atividade foi configurar uma máquina virtual com Suse Linux Enterprise mainframe para testar o funcionamento do mono e do WebISS.

Preparação do Ambiente

O Suse linux Enterprise foi instalado pela Megadata, e devido a questões de segurança suas partições foram divididas em diversos volumes. Para um ambiente de produção em ambiente de servidores isso é altamente recomendável, segue abaixo o particionamento utilizando no Linux no Mainframe.

/dev/disk/by-path/ccw-0.0.0200-part1 / ext3 acl,user\_xattr 1 1

/dev/vg00/lv06 /SUPORTE ext3 acl,user\_xattr 1 2

/dev/disk/by-path/ccw-0.0.0100-part1 /boot ext3 acl,user\_xattr 1 2

/dev/vg00/lv04 /home ext3 acl,user\_xattr 1 2

/dev/vg00/lv03 /opt ext3 acl,user\_xattr 1 2

/dev/vg00/lv05 /tmp ext3 acl,user\_xattr 1 2

/dev/vg00/lv01 /usr ext3 acl,user\_xattr 1 2

/dev/vg00/lv02 /var ext3 acl,user\_xattr 1 2

/dev/dasdg1 swap swap defaults 0 0

/dev/dasdh1 swap swap defaults 0 0

proc /proc proc defaults 0 0

sysfs /sys sysfs noauto 0 0

debugfs /sys/kernel/debug debugfs noauto 0 0

devpts /dev/pts devpts mode=0620,gid=5 0 0

/dev/dasdae1 /cdrom ext3 acl,user\_xattr 1 2

/dev/dasdaf1 /backup ext3 acl,user\_xattr 1 2

Considerando que este servidor não é um servidor de produção, mas um servidor que será usado para teste e compilação de pacotes do mono. A organização das partições não facilitou muito o trabalho, pois era necessário copiar código fonte de um lado para o outro. Sugere-se que caso seja necessário uma nova instalação de teste que sejam usadas no máximo 2 partições.

Com o servidor instalado e operacional foram iniciados os teste com o Z10 fazendo comparativos de desempenho, para verificar se não existia uma diferença muito grande de processamento entre as maquinas testadas, ou se seria necessário a criação de algum indexador para os testes.

Testes iniciais com Z10

A primeira coisa que precisamos é saber se o poder de processamento do mainframe na VM esta compatível com o PC que vamos usar para teste. Eles precisam ter uma poder de processamento parecido, senão nossos testes serão inúteis e precisamos de uma comprovação disso, fora do ambiente do mono ou de qualquer VM. Não queremos testar o hardware e sim o software que vai rodar nele. A idéia é saber se o mainframe é mais rápido ou mais lento para executar o Mono/.net.

# Bogomips

A primeira idéia que tivemos foi usar do Bogomips do kernel do Linux para saber se estão compatíveis. BOGOMIPS é uma combinação de Bogus e Mips. MIPS significa Milhões de Instruções por Segundo (Millions of Instructions per Second), ou Indicador sem importância de velocidade do Processador (Meaningless Indication of Processor Speed). O número que aparece na hora do boot é resultado de uma calibração de tempo do kernel, usado para curtíssimos atrasos de loops por alguns módulos de dispositivos.

Segundo De Lars Wirzenius: *`MIPS é a sigla para Milhões de Instruções por Segundo; ela é a medida para a velocidade da computação de um programa. Como a maioria destas medidas, ela é mais freqüentemente aumentada do que usada corretamente (é muito difícil comparar o MIPS para diferentes tipos de computadores). BOGOMIPS é uma invenção do Sr. Linus. O kernel (ou era um controlador de dispositivo?) precisa de um tempo de loop (o tempo também é uma sigla e/ou precisa ser exato para um método de espera de um loop sem uso), que deve ser calibrado para a velocidade do processador da máquina. Daí, o kernel mede na hora da inicialização quão rápido um certo tipo de loop roda em um computador. "Bogo" vem de "bogus", isto é, algo que é falso. Então, o valor do Bogomips fornece alguma indicação da velocidade do processador, mas é também um modo não muito exato para ser chamado de tudo menos Bogomips.*

De acordo com o mini-HOWTO do BOGOMIPS, a velocidade de sua máquina deve ser:

Common BogoMips Ratings

Processor BogoMips Comparison

--------- -------- ----------

Intel 8088 clock \* 0.004 0.02

Intel/AMD 386SX clock \* 0.14 0.8

Intel/AMD 386DX clock \* 0.18 1 (definition)

Motorola 68030 clock \* 0.25 1.4

Cyrix/IBM 486 clock \* 0.34 1.8

Intel Pentium clock \* 0.40 2.2

Na medição atual realizada temos:

* Para o AMD Atholn 64 x2 dual Core Processor 4400+ : 4620.98 Bogomips
* Para o IBM/S390: 1058bogomips

# Calculo de PI

2) A segunda idéia que eu tive é usar um código de C ou C++ compilado com gcc usando por exemplo um algoritmo de calculo de PI , com varias casa decimais. Posso compilar para código nativo em cada máquina e cronometrar o tempo de execução. A idéia que eles sejam similares para sabermos que as máquinas estão compatíveis no poder de processamento.

Calculo de PI simples o fonte pode ser encontrado em http://bellard.org/pi/pi.c

Para compilar no Windows instalar o Unix for service disponível no painel de controle do Windows. (<http://suacommunity.com/sua.aspx>) Para compilar com o gcc usar o seguinte comando:

# gcc –o pi.exe pi.c -lm

Código fonte do cálculo do PI usado está no apêndice deste documento.

Script para tomada de tempo no Linux :

#!/bin/bash

# primeiro teste de PI - calculo simples

# Saida arquivo time.txt

/usr/bin/time -o time.txt -a -f %e ./pi 1

/usr/bin/time -o time.txt -a -f %e ./pi 10

/usr/bin/time -o time.txt -a -f %e ./pi 100

/usr/bin/time -o time.txt -a -f %e ./pi 1000

/usr/bin/time -o time.txt -a -f %e ./pi 10000

/usr/bin/time -o time.txt -a -f %e ./pi 100000

Script para tomada de tempo no Windows. O comando time funciona um pouco diferente:

time ./pi.exe 1

time ./pi.exe 10

time ./pi.exe 100

time ./pi.exe 1000

time ./pi.exe 10000

time ./pi.exe 100000

|  |  |  |  |
| --- | --- | --- | --- |
| PI | IBM/S390 | AMD Athlon(tm) 64 X2 | % AMD mais rápido |
| 100 | 0,03 | 0,01 | 66,66667 |
| 1000 | 2,04 | 0,72 | 64,70588 |
| 10000 | 179,29 | 56,61 | 68,42546 |
| 100000 | 15931,09 | 4953,93 | 68,90401 |

# Compactação de Arquivos

3) Outra idéia é usar um compactador tipo RAR .Os algoritmos de compressão são bons testes de performance são altamente randômicos ocupam boa memória usam bastante processamento. Não tem cache que resista acabam testando o hardware diretamente. Quanto menos compressível o arquivo, melhor, pois o compressor vai se matar para tentar comprimir, passando por todos os passos do algoritmo, e não vai conseguir.

|  |  |  |  |
| --- | --- | --- | --- |
|  | IBM/S390 | AMD Athlon(tm) 64 X2 | % AMD mais rápido |
| Compactação | 16,43 | 9,8 | 40,35301 |

4) Uso do hdparm para testes. O hdparm é um utilitário muito usado, que permite ativar otimizações para o disco e medir velocidade. Não se trata necessariamente de um utilitário para "turbinar" o seu sistema, mas para descobrir e corrigir problemas de configuração do disco rígido que podem comprometer o desempenho.

Teste de escrita no Cache. Teste com hdparm para verificar o IO e velocidade de disco  
hdparm -T /dev/sda

|  |  |
| --- | --- |
| AMD  Timing cached reads:   2026 MB in  2.00 seconds = 1013.09 MB/sec Timing cached reads:   1962 MB in  2.00 seconds = 981.03 MB/sec Timing cached reads:   2036 MB in  2.00 seconds = 1018.22 MB/sec Timing cached reads:   2024 MB in  2.00 seconds = 1012.14 MB/sec Timing cached reads:   2050 MB in  2.00 seconds = 1025.12 MB/sec Timing cached reads:   2034 MB in  2.00 seconds = 1016.97 MB/sec  Z10 Timing cached reads:   468 MB in  2.01 seconds = 232.44 MB/sec Timing cached reads:   6218 MB in  2.00 seconds = 3111.11 MB/sec Timing cached reads:   6256 MB in  2.00 seconds = 3129.82 MB/sec Timing cached reads:   6104 MB in  2.00 seconds = 3053.78 MB/sec Timing cached reads:   6194 MB in  2.00 seconds = 3099.54 MB/sec Timing cached reads:   6220 MB in  2.00 seconds = 3112.44 MB/sec |  |

Teste de Escrita no Device hdparm -t /dev/dasda

|  |  |
| --- | --- |
| Z10 webissh:~ # hdparm -t /dev/dasda  Timing buffered disk reads:  110 MB in  3.05 seconds =  36.05 MB/sec  Timing buffered disk reads:  168 MB in  3.00 seconds =  55.94 MB/sec  Timing buffered disk reads:  200 MB in  3.04 seconds =  65.79 MB/sec  Timing buffered disk reads:  218 MB in  3.06 seconds =  71.22 MB/sec  Timing buffered disk reads:  228 MB in  3.03 seconds =  75.20 MB/sec  Timing buffered disk reads:  232 MB in  3.01 seconds =  77.11 MB/sec  AMD linux:/home/binhara/scriptZ10 # hdparm -t /dev/sda  Timing buffered disk reads:  304 MB in  3.00 seconds = 101.22 MB/sec  Timing buffered disk reads:  304 MB in  3.01 seconds = 101.13 MB/sec  Timing buffered disk reads:  300 MB in  3.00 seconds =  99.92 MB/sec  Timing buffered disk reads:  300 MB in  3.02 seconds =  99.44 MB/sec   Timing buffered disk reads:  298 MB in  3.01 seconds =  98.89 MB/sec  Timing buffered disk reads:  296 MB in  3.02 seconds =  98.14 MB/sec |  |

# Teste Ruby Destrossator

Esse script foi construido para o grupo Buscapé com o objetivo de gerar grandes volumes de dados aleatórios, para fazer teste de carga nos sistema de recomendação do grupo. As bases de dados são em arquivos texto e o ruby foi executado usando a máquina virtual java com Jruby.

O objetivo de roda o ruby no Java é garantir as escalabilidade, podendo usar multiplos processadores gerando código nativo para as máquinas.

Executando no PC:

linux:/home/binhara # time ./jruby-1.6.4/bin/jruby destrossator\_nowrite.rb

Finito (took 0.0 weeks, 0.0 days, 0.0 hours, 29.0 minutes, 17.1980000000001 seconds)

Executando no Z10:

webissh:/home/mono # time /root/jruby-1.6.4/bin/jruby destrossator\_nowrite.rb

Finito (took 0.0 weeks, 0.0 days, 0.0 hours, 31.0 minutes, 25.5609999999999 seconds)

Podesse verificar que o tempo demandado foi similar em ambas as plataformas.

Instalação Linux Z10

# Instalando Z10

Após a instalação do Linux no Z10 com sua versão especifica foram instalados as pacotes do Mono Extension. Mas se percebeu que a versão empacotada do mono era muito antiga e seria necessários recompilar o mono apartir dos fontes com uma versão mais atual.

# Instalando Pacotes Mono Extension

SUSE Linux Enterprise Mono Extension, um framework de aplicações. NET, permite que as organizações para executar seus Microsoft. NET servidor baseado em Linux. Baseado em Mono-uma fonte aberta, multi-plataforma. Âmbito e aplicação NET executado em conjunto com o SUSE Linux Enterprise Server. Ele permite que as organizações de TI facilmente porta. NET aplicações baseadas no Windows para o Linux sem um grande investimento em reescrever código. Com esta oferta, os desenvolvedores corporativos e ISV podem desenvolver novo Linux e aplicações multi-plataforma usando. NET enquanto recebendo atualizações de produtos, segurança e correções de bugs e suporte premiado SUSE.

Confira esta avaliação gratuita do SUSE Linux Mono Extension Enterprise, que inclui todo o conteúdo da versão completa mais 60 dias de patches e atualizações (ou 180 dias para System z). Isso não inclui o acesso a qualquer suporte técnico ou para qualquer atualização de software para além do período de avaliação. Para obter suporte técnico e acesso anual para correções e atualizações.

A instalação do Mono Extension é necessária para se obter os pacotes de mono compilados para o Mainframe, sem eles é impossível fazer o bootStrap da máquina virtual e com isso iniciar o processo de atualização do Mono.

# Atualização do Mono Apartir dos Fontes

A última versão do Mono portada e suportada pela Novell para o Mono no Mainframe é a versão 2.0. Uma versão muito antiga se comparada com a versão atual, desde a versão 2.0 foram lançadas oficialmente as versões 2.2, 2.4, 2.8, 2.10 e está em desenvolvimento a versão 2.11.

Realizamos o download do repositório do mono da versão 2.11 para fazer o processo de atualização do mono. Ao tentar compilar a versão 2.11 com a versão 2.0 do mono, obtivemos a mensagem que não era possível e que seria necessário usa no mínimo a versão 2.11.

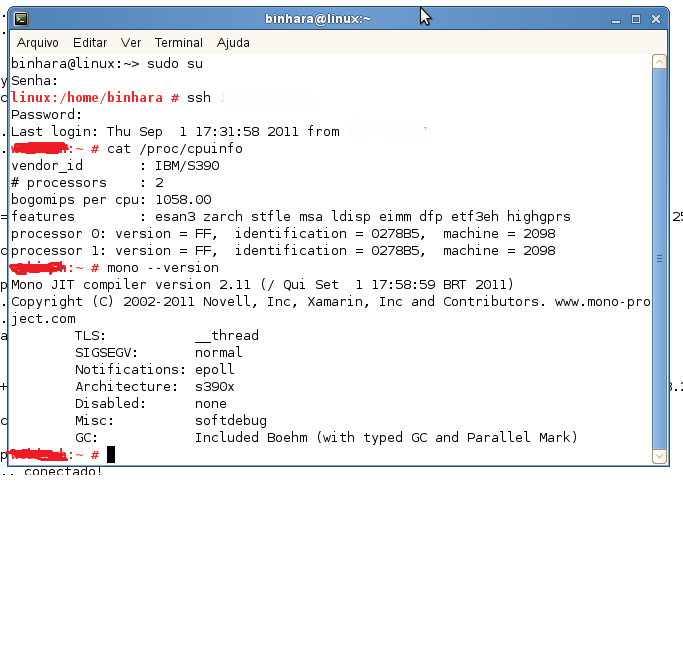
BUG JIT Mono

Compilação de versão

Foi necessário compilar versão após versão até chegar até versão 2.8 que era a versão necessária para atualizar para o Mono 2.11. Mas existe um bug no JIT do Mono s390 que não permite a compilação do mono na versão 2.11. A solução que conseguimos elaborar foi compilar como o mono 2.8 apenas a máquina virtual 2.10, e só então foi compilado a máquina virtual do mono 2.11 para o Z10. As bibliotecas do mono 2.11 foram compiladas em plataforma baixa e copiadas para o mainframe resolvendo o problema da geração dos binários para a plataforma Z10. Depois de realizado todo esse processo tem o mono 2.11 funcionando perfeitamente no ambiente Mainframe.

O código fonte do mono ocupa atualmente cerca de 300M bytes de espaço em disco. É uma quantidade de código fonte muito considerável certa de 8milhões de linhas de código. O processo de compilação cerca de 8% mais rápido que no AMD 64.

Após todo o processo de compilação ter sido executado e as bibliotecas compiladas no AMD terem sido copias para os locais adequados no Suse Linux é hora de testar e verificar se a VM e suas ferramentas estão funcionando adequadamente. Basta executar ***mono –version*** e teremos as informações de qual mono está funcionando.



# Instalação e Configuração do Apache

Com o mono instalado fica tudo mais fácil. Ao tentar colocar o mono funcionando com o apache recebemos uma mensagem de erro que a versão do módulo mod\_mono não era compatível. O problema que o módulo do mod\_mono é vinculado ao apache. Como não existe suporte para esta versão do mono foi necessário recompilar o mod\_mono. Mas para que ele funcione de forma adequada no Apache foi necessário recompilar o Apache também. Um processo bem trabalhoso, mas que funcionou bem no final.

Um dos grandes problemas que dificultaram o processo de instalação foi a falta de muitos pacotes compilados para o s390. A quantidade de pacotes de programas pronto para uso é muito pequena. Da mesma forma como precisamos compilar o Apache e o Mono, tivermos que compilar uma série de bibliotecas e programa necessários para o trabalho. Um exemplo que pode dar uma idéia da falta de pacotes, não existia um pacote do GIT pronto, uma ferramenta básica no processo de download do fonte do mono. Sem ela não poderíamos sequer baixar o código fonte do mono para iniciar os trabalhos. Essa falta de suporte gerou bastante trabalho que em outras plataformas seria trivial por já estar pronto.

Não conseguimos por exemplos compilar o FAST-CGI para testar em conjunto com o Apache e o mono. Ele apresentou uma serie de problemas de bibliotecas desatualizadas necessárias para a compilação. Com isso deixamos o mesmo de lado e não foi possível rodar o WebIss no Apache com FastGCI. Isso poderia ter um aumento significativo de performance do mono para prover aplicações web.

Compatibilidade do WebISS

## Compatibilidade de Código

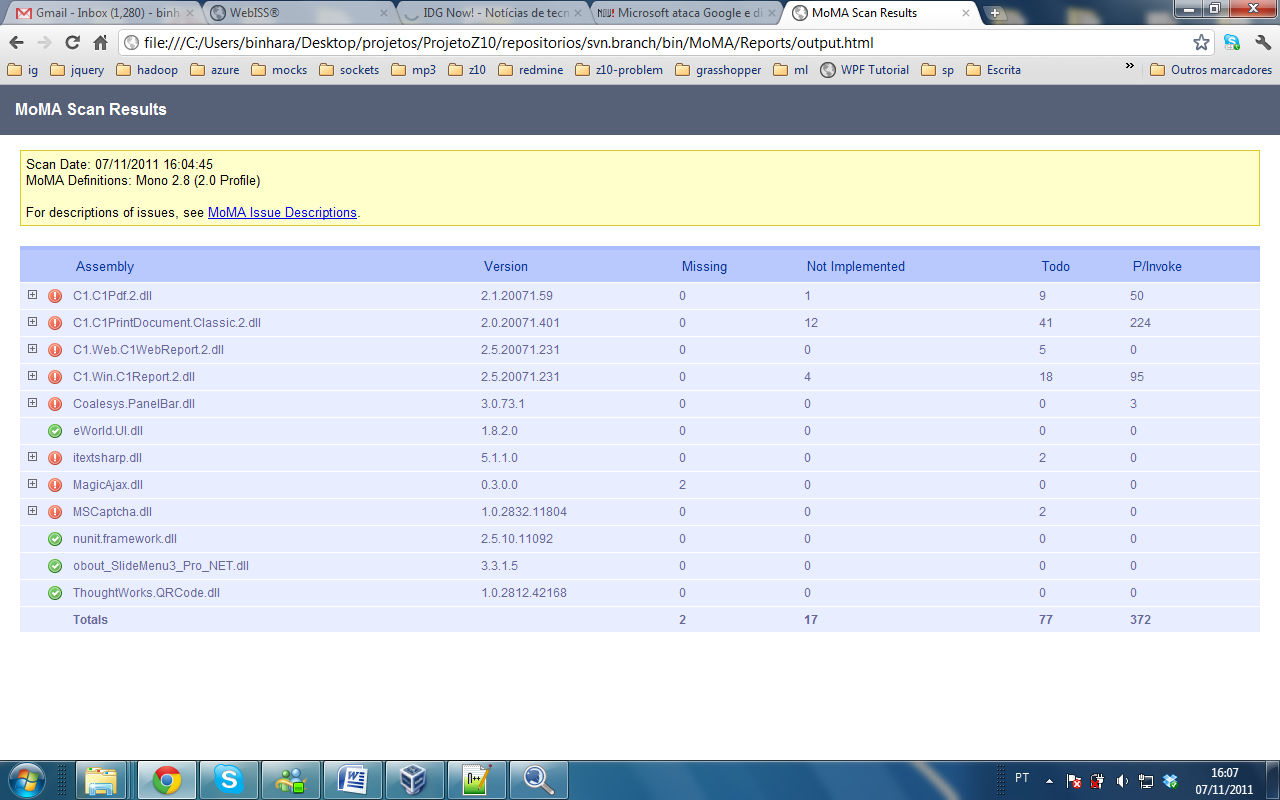
Atualmente o Mono suporta o código C# desenvolvido no WebISS, pois ele está escrito em .NET 2.0. O que precisa ser analisado como compatibilidade são bibliotecas externas ao .NET usadas no WebISS. O Mono é capaz de executar qualquer código .NET, mas não é capaz de executar código Windows. Para detectar códigos Windows em aplicações .NET o mono dispoem de uma ferramenta chamada Moma <http://www.mono-project.com/MoMA>



Está ferramenta é capaz de analizar os binários de uma aplicação em mostrar em que pontos de código existem uma chamada Windows. Essa ferramenta foi executada nos binários e nas bibliotecas do WEbISS e com isso foram mapeadas as possíveis incompatibilidades do WebIss.

## Compatibilidade de Bibliotecas

Usando a ferramenta MoMa foi possivel gerar um relatório de compatibilidade das bibliotecas. O relatório completo pode ser encontrado em RelatorioWebissCompatibilidadeDLL.html ou o resumo pode ser visto abaixo.



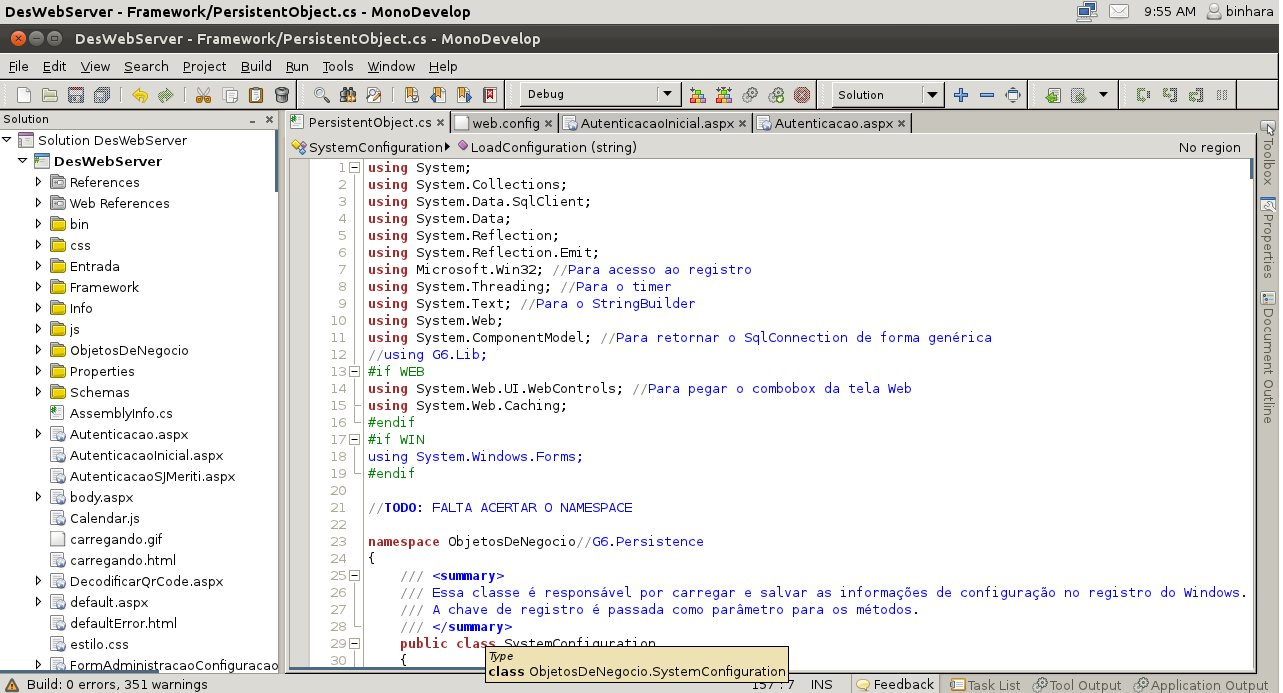
As bibliotetecas Nunit,obout\_slideMenu,ThoughtWorks, e World são 100% compatíveis com o mono.

As bibliotecas iTexsharp, MagicAjax MScaptcha podem funcionar parcialmente pois não fazem chamadas ao Windows.

Já as bibliotecas relacionadas a relatórios não irão funcionar pois apresentam muitas chamadas a API do Windows e é necessários remove-las e substitui-las por uma biblioteca compatível.

## Compilando o WebISS no Linux

Com o auxilio do mono Develop foi possivel compilar o WebISS no Linux. Revemos todas as bibliotecas incompatíveise e recebemos cerca de 40 Erros de compilação. Os erros erram gerados pricipalmente por erros de página de código.



## Formulários com Erros

Como removemos o componente de relatórios tivemos que remover os formularios com scripts incompatíveius. Foi realizada a remoção do Formularios com Relatorios com o seguinte script:

* FormCertidaoNegativa.aspx
* FormImpressaoGuiaLote.aspx
* FormLivroFiscal.aspx
* FormTransferirCredito.aspx
* FormPrintNFEAnonimo.aspx
* FormVerificarCreditoIptu.aspx
* G6RelatorioManual.cs
* objRelatorioGuia.cs
* ObjRelatorioGuiaAracaju.cs

## Formulários que contem MagicAjax

A Biblioteca magicAjax também se mostrou inconsistente e foi necessários remover os scripts ajax dos seguintes formulários:

* FormAdministracaoConfiguracao.aspx,
* FormAIFInterno.aspx,
* FormAjustaQtdeProfSocCivil.aspx
* FormApuracao.aspx
* FormAssistente.aspx
* FormAssistenteRecadastramento.aspx
* FormAutoLancamento2.aspx
* FormAutoLancamento.aspx
* FormBaixaBancaria.aspx
* FormCalculoImpostos.aspx
* FormCeC\_Antigo.aspx
* FormCeC.aspx
* FormCeCAutonomo.aspx
* FormCeCAvulso.aspx
* FormCeCPJ.aspx
* FormCeCUsuario.aspx
* FormExtApuracaoServPrestados.aspx
* FormExtApuracaoServTomados.aspx
* FormGeradorXmlNfse.aspx
* FormInconsistenciasNFeI.aspx
* FormNFeI.aspx
* FormNFSe.aspx
* FormNFSeAvulsa.aspx
* FormNFSeNiteroi.aspx
* FormNumeroSeqPeriodo.aspx
* FormParcelamento.aspx
* FormParcelamentoDA.aspx
* FormParcelamentoSS.aspx
* FormProcessosBatch.aspx
* FormRANFS.aspx
* FormTratarDivergencias.aspx

## Compatibilidade do código Fonte WebIss

Foram encontradas algumas incompatibilidade no código fonte, mas todos os problemas foram resolvildos. As bibliotecas incompativeis foram removidas, mas foi possível gerar uma nota eletrônica com os ajustes realizados.

**Problemas de Acentuação no código Fonte**

Trocar código fonte acentuado foram realizados cerca de 60 ajuste de nomes de variáveis e componentes que estavam acentuados. Uma possibilidade é fazer um script que faça esse ajuste.

**cat arquivo | tr çãõáéíó caoaeio >arquivo.novo**

Foram encontrados poucos erros de acentuação que tonaram incompatível com o sistema de arquivos do linux. Ao tentar mudar a página de código no VisualStudio não funcionou gerando outros problemas. Então os caracteres acentuados foram substituidos por caracteres sem acento. Com essas alterações o MonoDevelop passou a compilar os fontes sem maiores problemas.

Exemplos de problemas encontrados de acentuação de código.

edObservação

/home/binhara/svn.branch/FormInconsistenciasNFeI.aspx.cs(39,39): Error CS1056: Unexpected character `�' (CS1056) (DesWebServer)

/home/binhara/svn.branch/FormInconsistenciasNFeI.aspx.cs(102,102): Error CS1525: Unexpected symbol `,', expecting `;' (CS1525) (DesWebServer)

/home/binhara/svn.branch/FormInconsistenciasNFeI.aspx.designer.cs(71,71): Error CS1056: Unexpected character `�' (CS1056) (DesWebServer)

CarregaPainelCadastroAutorizações

/home/binhara/svn.branch/FormPessoaABRASF.aspx.cs(43,43): Error CS1056: Unexpected character `�' (CS1056) (DesWebServer)

/home/binhara/svn.branch/FormCancelamentoDocPago.aspx.cs(27,27): Error CS1056: Unexpected character `�' (CS1056) (DesWebServer)

chkGerarCrédito

/home/binhara/svn.branch/FormCancelamentoDocPago.aspx.cs(28,28): Error CS0128: A local variable named `dito' is already defined in this scope (CS0128) (DesWebServer)

/home/binhara/svn.branch/FormCancelamentoDocPago.aspx.cs(28,28): Error CS0128: A local variable named `dito' is already defined in this scope (CS0128) (DesWebServer)

/home/binhara/svn.branch/FormCancelamentoDocPago.aspx.cs(44,44): Error CS0136: A local variable named `dito' cannot be declared in this scope because it would give a different meaning to `dito', which is already used in a `child' scope to denote something else (CS0136) (DesWebServer)

/home/binhara/svn.branch/FormCancelamentoDocPago.aspx.cs(38,38): Error CS1525: Unexpected symbol `dito' (CS1525) (DesWebServer)

pnlIncluirLancCorreção

/home/binhara/svn.branch/FormSuspensaoReativacaoGuia.aspx.designer.cs(78,78): Error CS1056: Unexpected character `�' (CS1056) (DesWebServer)

/home/binhara/svn.branch/FormSuspensaoReativacaoGuia.aspx.designer.cs(81,81): Error CS1519: Unexpected symbol `;' in class, struct, or interface member declaration (CS1519) (DesWebServer)

O resultado que ao executar o sistema algumas páginas apresentam erros de caracteres como na imagem abaixo.



## Problema de Execução na camada de Persistência

Encontramos um problema de execução na camada de persistência no método PersisteObject. public DataTable ProcessSql. Esse método é bem importante no sistema, pois processa as requisições SQL que são enviadas para o banco. Ele foi construído de forma bem genérica para suportar qualquer comando SQL. No entanto, existem no C# comando específicos para cada tipo de comando SQL. O Mono por ter uma implementação diferente da Microsoft e seguir os padrões publicados pela Microsoft, não se comportou da mesma forma e emitiu uma mensagem de erro. Fizemos um ajuste no método usando as implementações específica e a aplicação funcionou corretamente.

Método original:

1. **public** DataTable ProcessSql(**string** commandString)              {
2. **bool** localConnection = **false**;
3. **if**(sqlConnection == **null**)
4. {
5. **string** connectionString = GetConnectionString();
6. sqlConnection = new SqlConnection(connectionString);
7. sqlConnection.Open();
8. localConnection = **true**;
9. }
10. DataTable dataTable;
11. **try**
12. {
14. **SqlDataAdapter dataAdapter = new SqlDataAdapter(commandString, sqlConnection);**
15. **dataAdapter.SelectCommand.CommandTimeout = timeout;**
16. **SqlCommandBuilder scb = new SqlCommandBuilder(dataAdapter);**
17. **dataAdapter.SelectCommand.Transaction = sqlTransaction;**
18. **DataSet dataSet = new DataSet();**
19. **dataAdapter.Fill(dataSet);**
21. **if(dataSet.Tables != null && dataSet.Tables.Count > 0)**
22. **dataTable = dataSet.Tables[0];**
23. **else**
24. dataTable = **null**;
26. }
27. **catch**(Exception e)
28. {
29. **if**(e.GetType().Name.ToLower() == "sqlexception")
30. {
31. }
32. **finally**
33. {
34. **if**(localConnection)
35. {
36. sqlConnection.Close();
37. sqlConnection = **null**;
38. }
39. }
40. **return** dataTable;

Alteração no método para funcionamento e compilação correta

1. **public** DataTable ProcessSql(**string** commandString)              {
2. **bool** localConnection = **false**;
3. **if**(sqlConnection == **null**)
4. {
5. **string** connectionString = GetConnectionString();
6. sqlConnection = new SqlConnection(connectionString);
7. sqlConnection.Open();
8. localConnection = **true**;
9. }
10. DataTable dataTable;
11. **try**
12. {
13. **if**(commandString.StartsWith("DELETE"))
14. {
15. SqlCommand cmd = new SqlCommand(commandString, sqlConnection);
16. cmd.Transaction = sqlTransaction;
17. cmd.ExecuteNonQuery();
18. dataTable = **null**;
19. }
20. **else**
21. {
23. SqlDataAdapter dataAdapter = new SqlDataAdapter(commandString, sqlConnection);
24. dataAdapter.SelectCommand.CommandTimeout = timeout;
25. SqlCommandBuilder scb = new SqlCommandBuilder(dataAdapter);
26. dataAdapter.SelectCommand.Transaction = sqlTransaction;
27. DataSet dataSet = new DataSet();
28. dataAdapter.Fill(dataSet);
30. **if**(dataSet.Tables != **null** && dataSet.Tables.Count > 0)
31. dataTable = dataSet.Tables[0];
32. **else**
33. dataTable = **null**;
34. }
35. }
36. **catch**(Exception e)
37. {

## Componentes de Relatórios

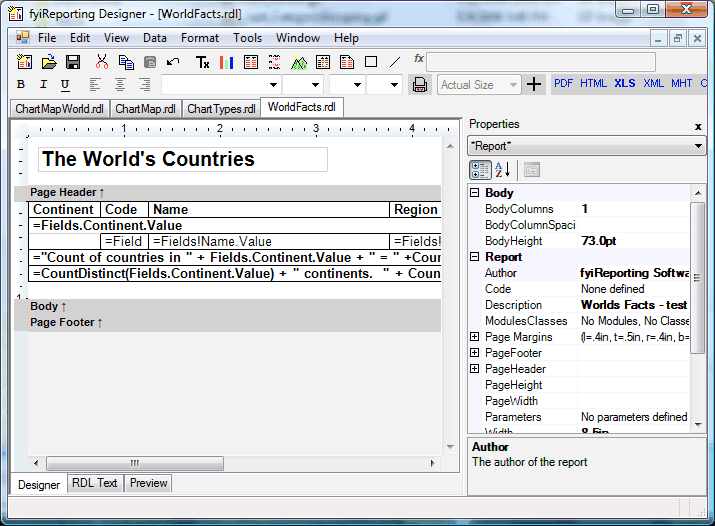
Um componente muito importante no sistema WebIss é o gerador de notas fiscais em PDF. No caso do WebISS ele usa uma biblioteca externa para gerar as notas fiscais e outros documentos no formato de relatórios impressos.

O componente usando hoje utiliza a API do Windows e isso torna essa biblioteca incompatível com o mono. Como não se possue o código fonte dessa biblioteca, visto que é uma biblioteca de terceiro, não é possível fazer ajuste para tornar compativel com o MONO. O que se sugere neste caso é a troca por um componente opensource que seja 100% compatível com o mono.

Existe uma centena deles e alguns componentes de relatórios selecionados para o uso são:

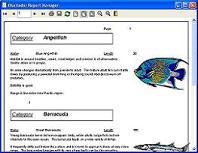
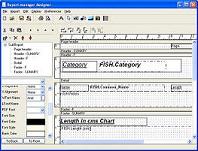
**Projeto RDL**

O Projeto RDL fornece uma implementação de qualidade da especificação RDL. A DLL compacto está disponível para incorporar o escritor de relatório em suas próprias aplicações. Um controle. Net permite-lhe incorporar relatórios dentro de seu visual aplicações e permitir a impressão. Uma biblioteca ASP.NET permite que você use seus relatórios em suas aplicações web. Um designer WYSIWYG está disponível para os desenvolvedores e não desenvolvedores para criar relatórios. O Projeto RDL é escrito em C # e requer .Net 2. <http://www.fyireporting.com/>



**Report Manager**

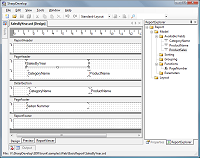
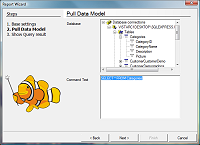
Report Manager é um aplicativo de relatório (Designer Gerenciador de Relatórios) e também um conjunto de componentes para Delphi, Builder e Kylix, também suporta ambiente de desenvolvimento aceitar controles ActiveX (Visual Basic, Visual FoxPro, Visual Studio.Net qualquer linguagem ...). C dinâmica biblioteca padrão com funções exportadas é fornecida para utilizar o motor com qualquer outra língua como o GNU C. <http://reportman.sourceforge.net/>



**SharpReports**

SharpDevelop Relatórios (SDR) é a fonte aberto. Transporte NET relatórios com SharpDevelop. SDR suporta Windows (WPF ou Windows Forms), bem como da Web (ASP.NET) aplicações graças à sua capacidade de usar impressoras ou saída diretamente para PDF. Bases de dados (não limitado ao SQL Server!), Bem como listas de objetos são suportados como fontes de dados.

http://sharpdevelopreports.net/default.aspx?AspxAutoDetectCookieSupport=1

Exemplo de código de geração de um relatório.

1. protected void Button1\_Click(object sender, EventArgs e)
2. {
3. contributors = CreateTestList();
4. // instance reporting engine
5. // assign parameters
6. ReportEngine engine = new ReportEngine();
7. string reportPath = Server.MapPath("ContributorList.srd");
8. ReportModel reportModel = ReportEngine.LoadReportModel(reportPath);
9. PageBuilder pageBuilder = engine.CreatePageBuilder(reportModel, contributors);
10. pageBuilder.BuildExportList();
11. string outputPath = Server.MapPath("ContributorList.pdf");
12. // render report
13. PdfRenderer pdfRenderer =
14. PdfRenderer.CreateInstance(pageBuilder.Pages, outputPath, false);
15. pdfRenderer.Start();
16. pdfRenderer.RenderOutput();
17. pdfRenderer.End();
18. // send report to the client
19. Response.ContentType = "Application/pdf";
20. Response.WriteFile(outputPath);
21. Response.End();
22. }

## Considerações Sobre Compatibilidade

O Sistema WebIss não foi pensando para rodar sobre plataforma linux, mesmo assim, eles esta funcionando bem, com exceção dos relatórios.

Apenas com uma boa orientação técnica e usando componentes 100% .NET, a própria evolução natural do WebIss poderá rodar no Linux 100% sem nenhum ajuste ou mudança no código.

O maior trabalho a ser realizado atualmente é a troca do componente de relatórios.

É visível a diferença de performance na WEB entre o AMD e o Z10 rodando a mesma versão.

Instalando o WebISS

Deploy do WEBIIS no Linux

O processo de deploy consiste em instalar o sistema WebIss no Linux. Para garantir a compatibilidade da aplicação com o mono a aplicação foi toda recompilada no MonoDevelop que é a IDE de desenvolvimento de aplicativos mono para linux.

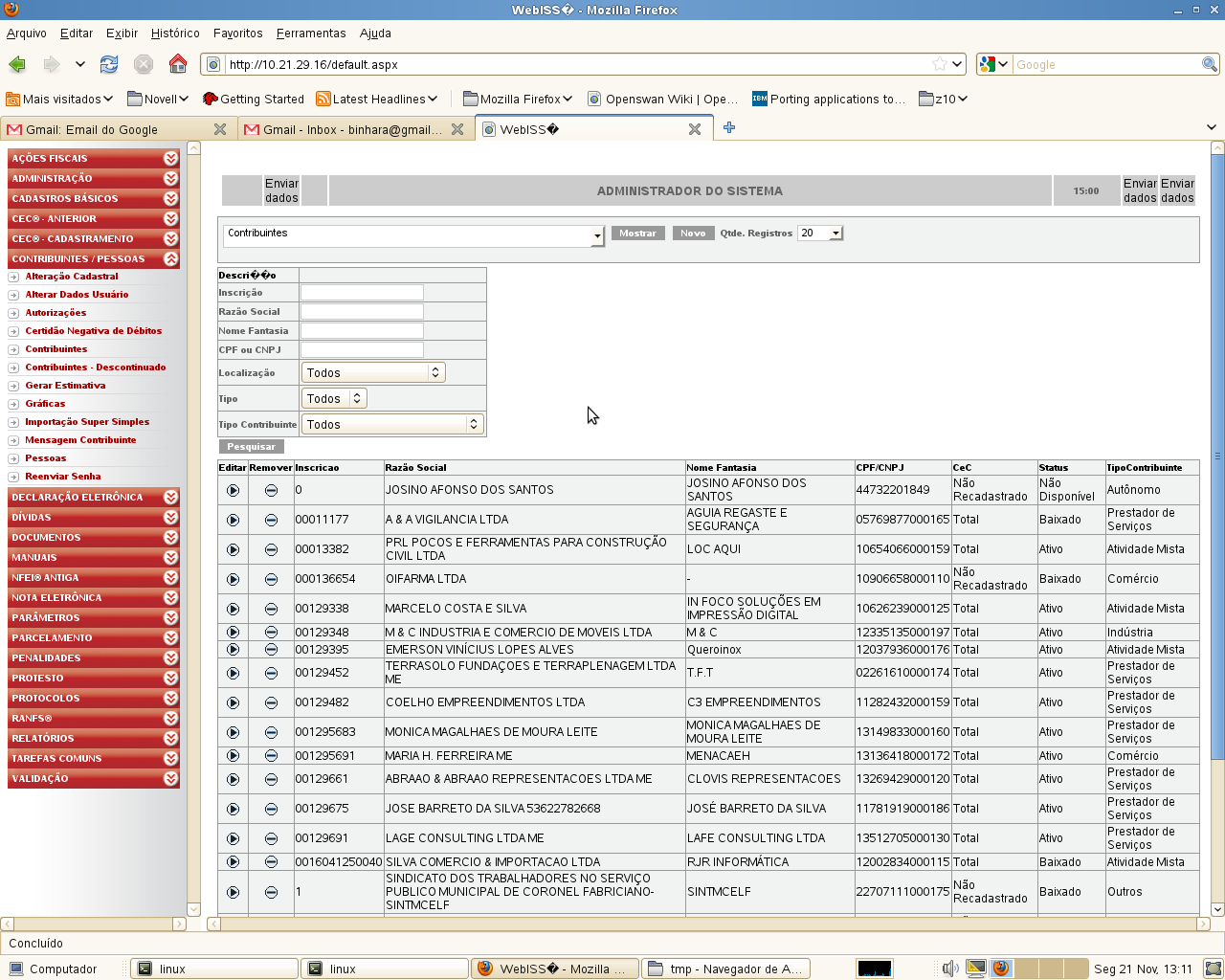
Basicamente o MonoDevelop foi instalado no SuseLinux para compilar o WebISS. Foi copiada a solução completa do Visual Studio dentro do linux, e foi aberta a solução do Visual Studio no MonoDevelop sem nenhum problema de compatibilidade.

Ao compilar a solução foram alguns problemas que foram descritos na continuação desde documento. Com todos os problemas resolvidos for gerada os binarios de de instalação do aplicativo.

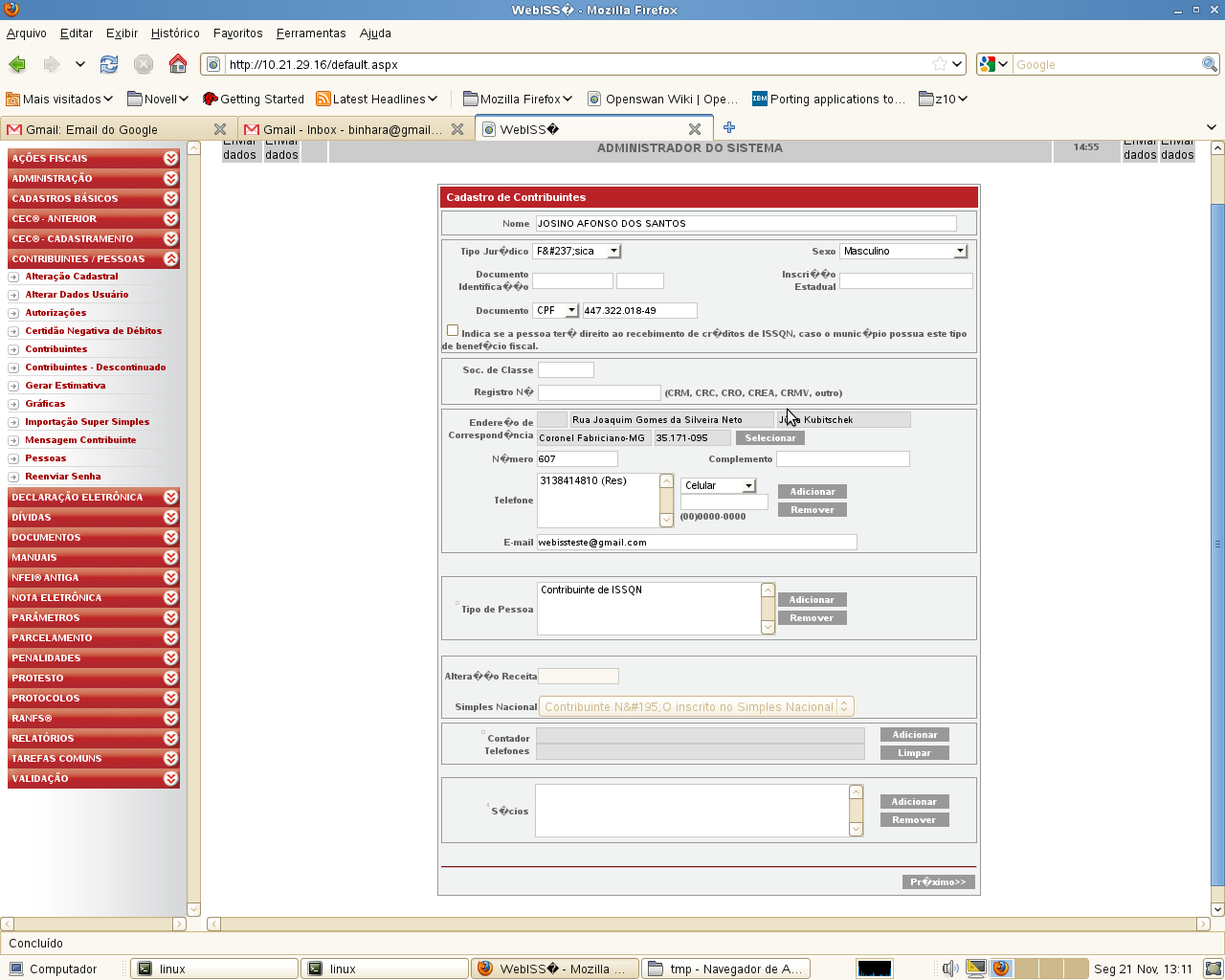
Bastou copiar os arquivos binários no diretório do apache e automaticamente o modulo do mono reconheceu a aplicação e foi possivel acessá-la na web pelo Apache.

Deploy do WEBIIS no Linux Z10

Para fazer o deploy no Z10 foi necessária a recompilação do apache com o módulo atualizado no modMono. Foi copiado os binários do WebIss gerados pelo MonoDevelop para o diretório do pache no Z10. A aplicação rodou com sucesso no Z10 como pode ser visto na imagem abaixo.

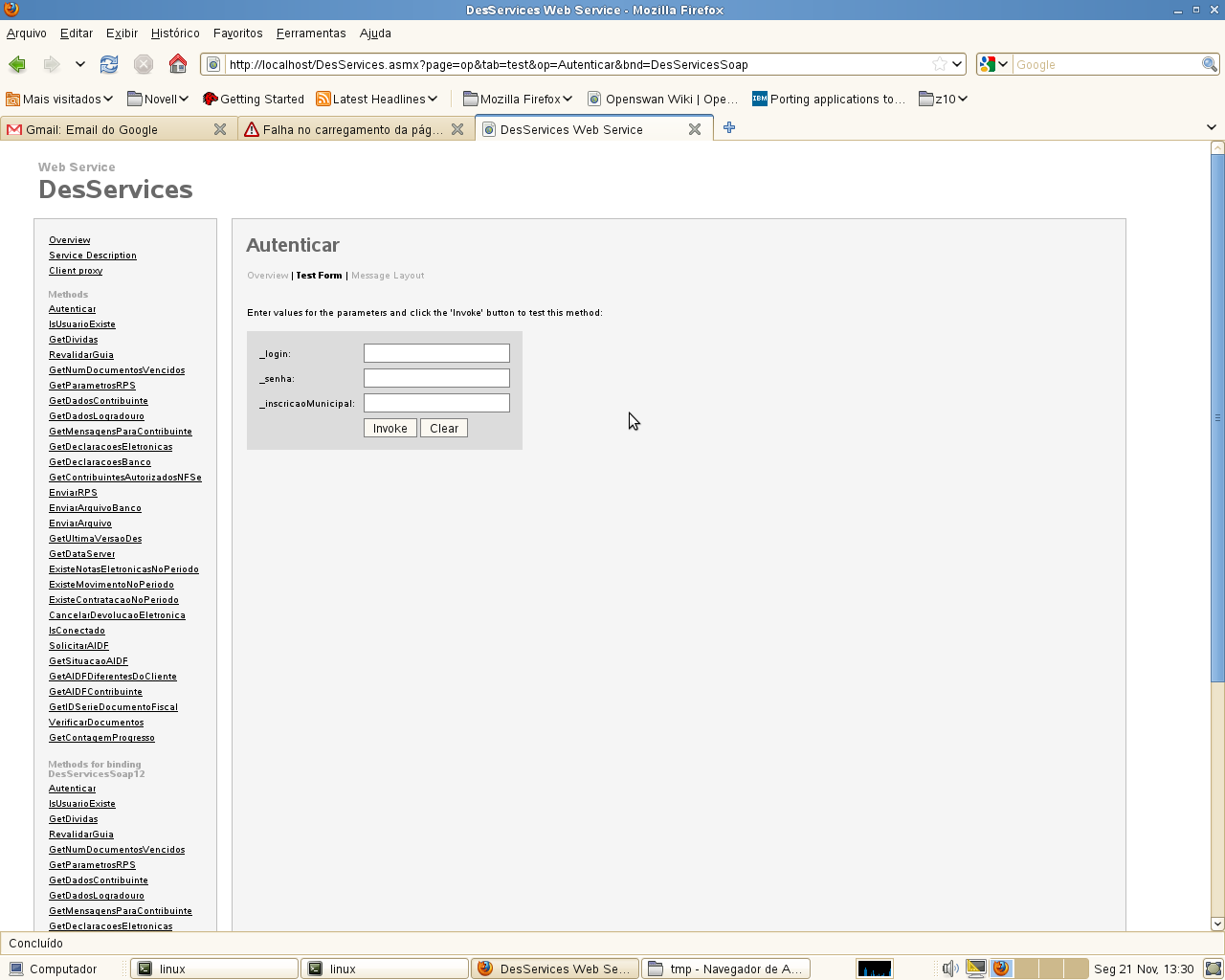


Não foi necessária nenhuma configuração especial no apache para a execução da aplicação.



Deploy dos Web Services - DesWebService

WebService estão bem suportado no Mono. Não tivemos problemas para compilar ou fazer o deploy da web serviço no Apache. Eles não usam nenhuma biblioteca externa ou qualquer função não implementada no Mono. A seguir uma tela da web services rodando no mono com apache



Testes do WebISS

# Planejamento dos testes

Serão criados teste para atuar e fazer medições em múltiplo níveis. A idéia é conseguir um isolamento de cada teste para que estes não fiquem sujeitos a interferências externas. A aplicação WEBIss pode ser entendida em múltiplas camadas, que podem ser testadas de forma independente ou em conjunto.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Banco de Dados | MONO ou .NET | Camada de Persistência | Lógica de Negócio | ASP.NET WEB.Forms | Servidor WEB | REDE | Cliente Web Browser |

**Banco de dados:** É o servidor de banco de dados com suas tabelas podem ser o SQLServer ou DB2.

**MONO ou .NET:** É a máquina virtual que executa todo o aplicativo WEBIss dentro do sistema operacional. Em ambiente Windows é usada a VM CLR e no ambiente Linux é utilizado a VM Mono.

**Camada de Persistência:** É a camada que entende o banco e gera as consultas SQL que serão consumidas pela aplicação. Essa camada esconde os comandos SQL e entrega para a aplicação objetos projetos prontos que são usados diretamente. Está camada foi construída em C# pela benefix e permite que a aplicação possa trocar de banco facilmente.

**Lógica de Negócio:** Onde são encontradas as regras de funcionamento da aplicação.

**Asp.Net Web.Forms:** Componente da Microsoft responsável por gerar as telas HTML do sistema. É a interface do usuário propriamente dita.

**Servidor WEB:** Esse também é um componente a ser testado, pois a variação desse software ou a melhor o pior calibração desse software pode dar ganhos expressivos de performance. No caso será usando no Windows o IIS e no ambiente Linux serão testados o Apache e o Nigx.

**Rede:** é a camada que conecta o usuário ao servidor. Essa camada deverá ser isolada sempre que possível para evitar latência de rede.

**Cliente Web Browser:** São os navegadores Web, atualmente os diferentes engines de renderização tem causado grande impacto na percepção de velocidade das aplicações por parte dos usuários. É um componente a ser isolado nos teste também.

## Testes unitários

Teste de unidade é toda a aplicação de teste nas assinaturas de entradas e saídas de um sistema, consiste em validar dados válidos e inválidos via I/O (entrada/saída) sendo aplicado por desenvolvedores ou analistas de teste. Uma unidade é a menor parte testável de um programa de computador. Em programação procedural, uma unidade pode ser uma função individual ou um procedimento. Idealmente, cada teste de unidade é independente dos demais, o que possibilita ao programador testar cada módulo isoladamente.

Tem essa visão em mente serão construídos testes da aplicação que possibilitem a medição do tempo de execução de forma automatizada sem a intervenção humana. Possibilitando executar os mesmo testes diversas vezes. A fim de comprovar a performance do sistema em ambientes diferentes, com o mínimo de esforço humano.

Construção Teste unitários de desempenho

Estes testes deverão testar diversas situações de forma isolada sendo elas: Teste de acesso a banco SQL, Teste do Framework de Persistência, Teste da Lógica de negócio, Teste de Algoritmos Genéricos e Específico, Teste de Execução e uso de memória, Teste de partes Funcionais do sistema, Teste do Servidor Web, Teste de Integração, Teste de Interface Web , Construção Teste unitários via Selenium com o WebISS.

### Teste de acesso a banco SQL

Estes testes serão construídos para testas o acesso ao banco. Serão criadas duas Tabelas QUERY\_TEST e TRASACTION\_TEST com a seguinte estrutura:

BANCO TESTE

CREATE TABLE [dbo].[QUERY\_TEST](

[ID] [decimal](18, 0) NULL,

[NAME] [varchar](8) NULL,

[STRING1] [varchar](1000) NULL,

[BIGDECIMAL1] [decimal](18, 0) NULL,

[BIGDECIMAL2] [decimal](18, 0) NULL,

[DATE1] [date] NULL,

[DATE2] [date] NULL

) ON [PRIMARY]

CREATE TABLE [dbo].[TRANSACTION\_TEST](

[ID] [decimal](18, 0) NULL,

[NAME] [varchar](8) NULL,

[STRING1] [varchar](1000) NULL,

[BIGDECIMAL1] [decimal](18, 0) NULL,

[BIGDECIMAL2] [decimal](18, 0) NULL,

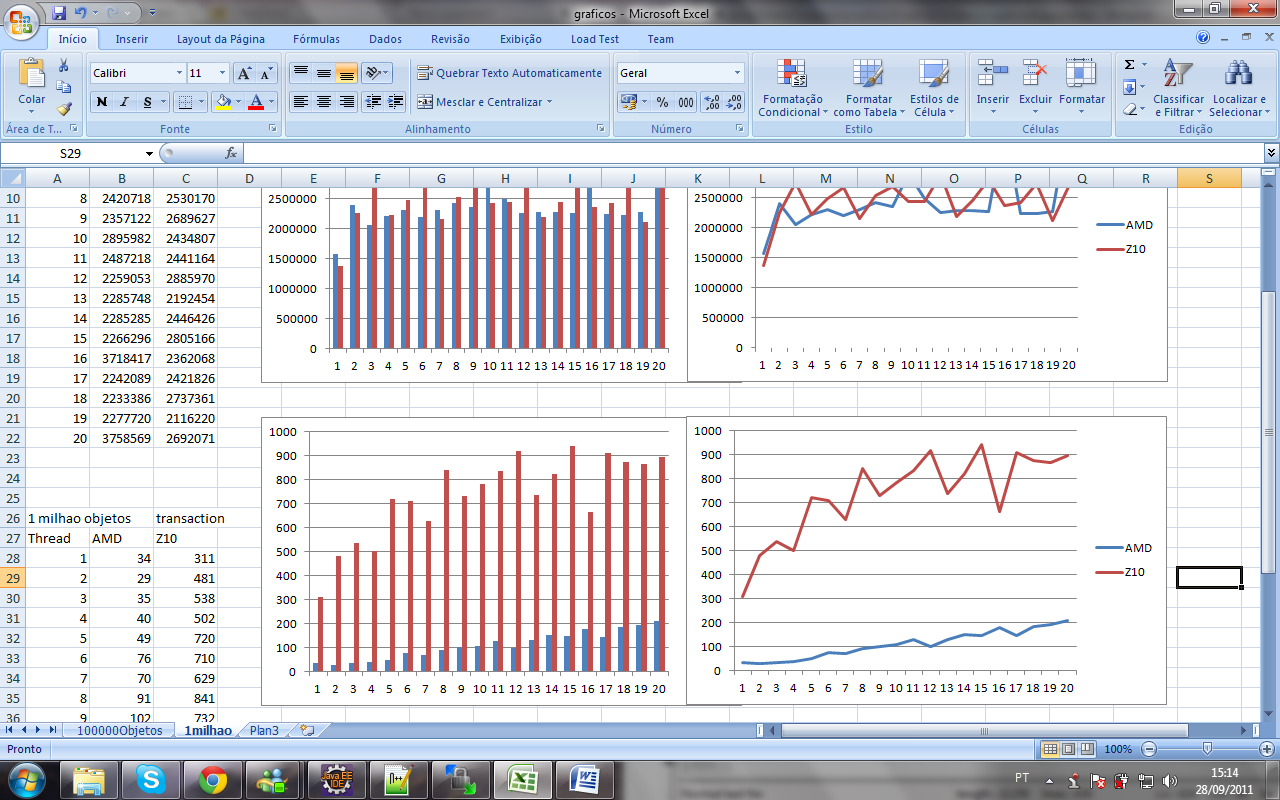
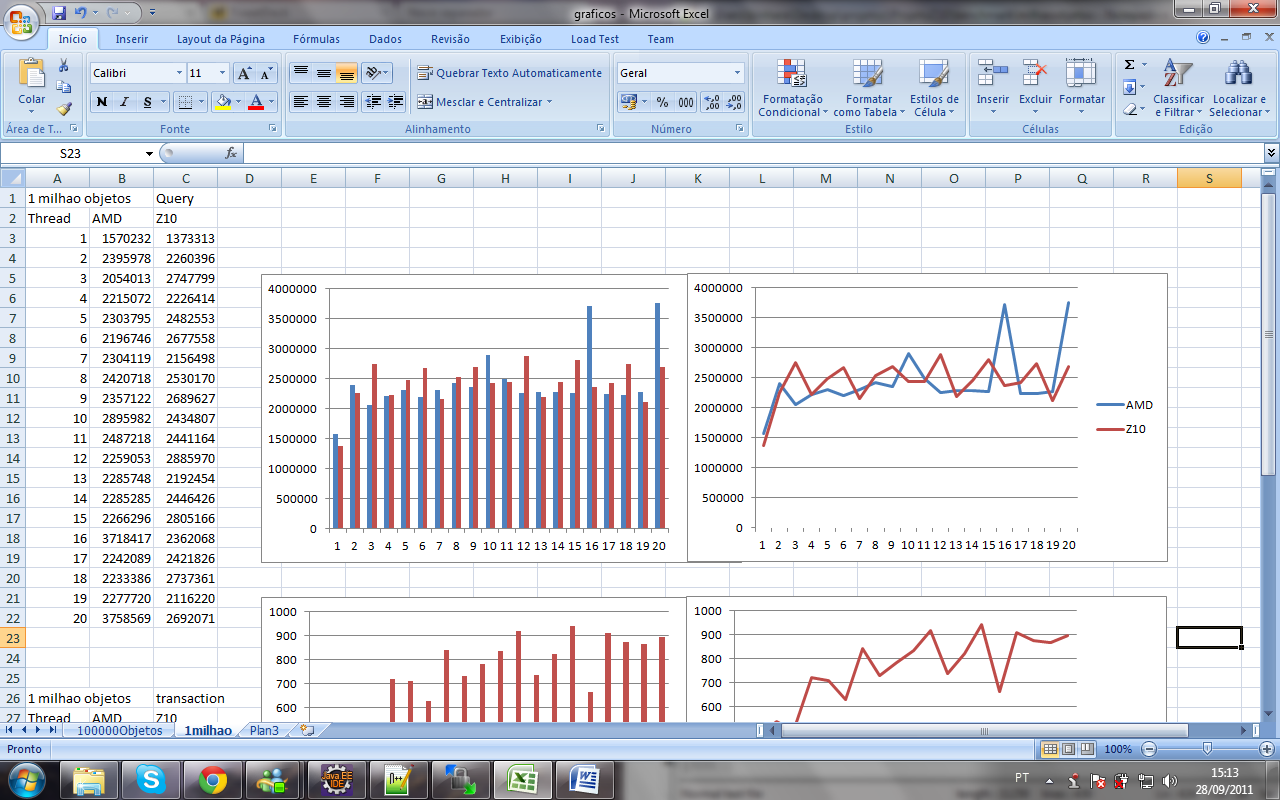
[DATE1] [date] NULL,

[DATE2] [date] NULL

) ON [PRIMARY]

1) Serão criados os objetos em memória, será executado o teste de consulta aos objetos em memória com múltiplas threads (1 a 20 threads). Resultados para Consultas e Transações em memória para 10mil objetos:

Para 1 milhão de Objetos:



## Teste de Algoritmos Genéricos

Foram executados alguns algoritmos genéricos para medir a performance da máquina virtual mono rodando nas duas plataformas de hardware em Linux e Windows. O teste consiste um número X de operações e é medido o tempo em milissegundos. Quanto menor o valor em milissegundos mais rápidos foi a operação. Foram feitos teste com manipulação de ArrayList, Strings, Soma de inteiros e ponto flutuante, exception e reflexão e recursão. Abaixo temos os números encontrados para a execução no ambiente Linux em AMD64, Linux em Z10 e Windows 2008 com .NET. Foi utilizando sempre a mesma versão de Linux com a mesma versão de mono, no caso mono 2.11.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Plataforma /Algoritmos | ArrayList | StringBuilder | Adição de Integer & Floating | Exception | Reflection e recursão |
| Mono Linux | 452 | 768 | 5192 | 321 | 4684 |
| Mono Z10 | 1038 | 1926 | 12164 | 543 | 9787 |
| dotNet Windows | 499 | 639 | 5194 | 3057 | 5865 |

Visualizando esses testes genéricos pode-se se perceber que o desempenho do mono em Intel esta similar a performance do .NET, mas pode-se ver que ele foi muito mais rápido que o .NET no lançamento de Exceptions isso se deve ao fato o tratamento de exceptions é mais simplificado no Mono, e não tem todas as checagens de CAS que tem no .NET. Outro ponto é que o mono se apresentou mais rápido que o .NET em reflection e recursão. Não encontramos nenhuma razão especial para essa situação esteja acontecendo. Pela conversa com os desenvolvedores do mono parece de dar pela eficiência da estutura do Linux, como gerenciamento de processos do kernel e memória, sem ter todas as camadas que o Windows tem.

O Mono em Intel está com performance similar a do .NET. O Mono está mais rápido que o .NET em Exceptions pois o tratamento de exceptions é mais simplificado no Mono, ele não tem todas as checagens do .NET. O Mono está mais rápido que o .NET em reflection e recursão, tal eficiência se dá pela estrutura do Linux ser mais eficiente que o Windows. A performance do Mono no Z10 foi bem pior que o Mono no AMD. Tendo em vista que o Linux e o Mono é o mesmo fonte utilizado. Tem-se um forte indício que a performance da aplicação na máquina Z10 é bem inferior que no AMD.

## Algoritmo multi thread

Para evitar questionamento sobre o teste com aplicações single thread, e também a capacidade da máquina virtual do Mono em executar aplicativos multi-thread. Forma usando teste em multithread na máquina virtual java.

No teste inicial foram criandos objetos simples em memória em centenas de milhares. E depois eram feitas consultas em memória e cronometrado o tempo de execução de cada tarefa. Para a quantidade de 1milhão de objetos os sistema na média ficaram empatados. AMD=2.426.000 Z10= 2.434.000 com 8mil operações por segundo de diferença.

Na sequencia foram realizar teste de IO, escrevendo os objetos de memória no disco usando processo de serialização de objetos. Em seguida era feitas consutlas, novamente para uma quantidade de 1 milhão de objetos.

No gráfico abaixo pode ser visto que o Z10 foi muito mais eficiente com IO .

Realizando um teste com consultas ao SQL SERVER. Tanto Z10 quando o AMD realizava consulta a um SQL Server em um servidor Windows. É visível o ganho em número de consultas usando o AMD. É interessante observar que com o aumento de número de threads paraleas o AMD teve um ganho de performance.

Considerando que a aplicação WebIss não fará uso de IO do mainframe, e o grande requisito do WebIss é processador, memória e acesso ao SQL Server. A aplicação WebIss terá uma eficiência muito maior rodando em uma arquitetura AMD/Intel que no Z10.

## Criação de Objetos em Memória

Depois dos teste com a máquina virtual java, foram iniciados os teste com C#. Usando .NET no windows e Mono no Linux (Z10 e AMD). Abaixo podem ser visto os resultados para criação de objetos em memória para volumes de 1mil, 10mil e 100mil objetos.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1mil objetos** | **10mil**  **objetos** | **100mil**  **objetos** |
| **Mono AMD** | 17 ms | 115 ms | 1059 ms |
| **Mono Z10** | 25 ms | 146 ms | 1458 ms |
| **.NET AMD** | 31 ms | 93 ms | 936 ms |

Pode ser observado que quanto menor é melhor, pois levou menos tempo para criar os objetos em memoria. O C# rodando em plataforma Windows foi nitidamente márápido que no mono no linux, mesmo rodando no ambiente MainFrame

## Teste SQL – 1 Thread .C#

No próximo teste foram criados 1mil objetos em memória, em seguida esses objetos foram salvos no banco de dados através de comandos SQL. Depois de criada a base de dados foram realizadas 1 mil buscas , 1 mil updates alterando os dados do bando de dados. A última etapa foi deletar os registros criados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Criar Memória | Insert | Busca | Update | Delete |
| Mono AMD | 17 | 8727 | 936 | 9472 | 11402 |
| Mono Z10 | 25 | 4337 | 959 | 4379 | 4149 |
| .NET AMD | 31 | 10077 | 1029 | 9406 | 9592 |

O interessante deste teste foi que o Z10 com mono foi mais rápido que os demais na hora de inserir, atualizar e remover registro do banco. Mas a velocidade na busca dos objetos foi similar, isso comprova a superioridade do Z10 em manipulação de disco e IO pois essas operações gastam muito mais tempo de IO que processamento e memória. Já nas busca, o banco de dados se encarrega de trabalhar com esses mecanismos, sendo usado mais processamento e memória. Neste caso o resultado foi uma performance equivalente nas três plataformas.

A mesma situação se repte quando foram executados os testes para a quantidade de 10mil registros, como pode ser visto no gráfico logo abaixo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Criar Memória** | **Insert** | **Busca** | **Update** | **Delete** |
| **Mono AMD** | 115 | 93232 | 26885 | 137954 | 131444 |
| **Mono Z10** | 146 | 43677 | 33916 | 94313 | 101117 |
| **.NET AMD** | 93 | 99637 | 29749 | 154284 | 153645 |

Testes dos WCF – Nota Eletrônica

Existe apenas um método não implementado no Mono. Mas este método é o método usado para autenticar as chaves digitais. Isso é um ponto vital para o funcionamento deste módulo do WebISS, sem ele esse módulo não funciona.

Opções são :

* Implementar a funcionalidade o Mono. É um pouco desaconselhado pois será preciso uma pessoa altamente especializada, e é um recurso não disponível na equipa da WebISS. Mas é possível contratar sob demanda o que pode resolver o problema de compatibilidade.
* Mudar a implementação da Aplicação WCF, pois existem diversos outros métodos no mono que fazem o trabalho de forma similar e já estão implementados e funcionando no mono. Essa pode ser uma solução simples e barata a ser implementada.
* Outra opção é migrar o WCF para WebServices, pois o WebISS não é utilizada nenhuma característica especial do WFC . O WebISS não perderia nada em termos de funcionalidade e o processo de migração não seria complicado.

Observação:

Com a evolução rápida do mono e com o aumento do uso de aplicativos mobiles que é o atual foco do mono, este método será muito importante no uso em aplicativos mobiles. Conseguente é possivel que nas próximas versões do Mono esse método esteja implementado. Deixando a camada de WFC 100% compatível com a atual versão do WebISS.

Considerações Finais

# Resultados Encontrados

A principal questão deste trabalho era saber se era possível executar o WebISS no linux num Mainframe Z10. Foi comprovado que SIM, é possível executar o WebIss no Z10 usando a plataforma Mono com mínimo de esforço de programação e testes.

Outro item do trabalho foi avaliar o esforço necessários do porte do WebISS para o Z10. Isso foi realizado com sucesso executando algumas adaptações já descritas:

* Troca do componente Ajax e Componentes de Relatórios.
* Conversão do componente escrito em WCF para WebService.
* Migração das Chaves de Criptografia do ISS para o Apache.

Foram também decritos os problemas encontrados e as possíveis soluções como:

* Necessidade de uma maior conjunto de Testes: Criação de testes automáticos que garantirão o correto funcionamento em ambas as plataformas.

Avaliar desempenho do Z10 em relação a plataforma baixa como servidor de aplicação Web

* O Z10 teve performance equivalente a um AMD de 5 anos atrás. Com certeza a aplicação não terá a mesma velocidade que tem usando a infraestrutura de processadores Xeon atuais.

Z10 não é tão performático como um INTEL/AMD para rodar uma aplicação Web como o WEBISS. Tudo que esta fora do ambiente do mainframe se torna lento, o Z10 não está preparado para esse tipo de situação. O Z10 não escala a aplicação sozinho, não possue nenhum mecanismo aparente para escalonamento da aplicação. A aplicação e o ambiente do sistema operacional precisam ser customizados para rodar nesse ambiente de forma a escalarem a quantidade de usuários.

Mesmo que seja migrado 100% o WebIss para o Z10 ele não terá a mesma performances que o atual ambiente de servidores windows. A única vantagem no uso do Z10 é seu tempo de disponibilidade, o que também é atingindo com a plataforma Windows hoje.

Muito provavelmente a aplicação no Z10 só terá ganho de performance sendo escrita de forma nativa para o z10. O mainframe não parece disponibilizar nenhum mecanismo de escalabilidade automática da aplicação. Seja Java ou C# , a aplicação deverá ter inerente a ela a escalabilidade para que novos servers possam ser criados para distribuir carga.

Tudo indica que será muito mais barato manter o sistema WebISS rodando em plataforma Windows que migrá-lo para Linux num MainFrame.

Código Fontes de Teste

# Código fonte do Programa de Cálculo de PI

1. #include <stdlib.h>
2. #include <stdio.h>
3. #include <math.h>
5. #ifdef HAS\_LONG\_LONG
6. #define mul\_mod(a,b,m) (( (long long) (a) \* (long long) (b) ) % (m))
7. #else
8. #define mul\_mod(a,b,m) fmod( (double) a \* (double) b, m)
9. #endif
11. */\* return the inverse of x mod y \*/*
12. int inv\_mod(int x,int y) {
13. int q,u,v,a,c,t;
15. u=x;
16. v=y;
17. c=1;
18. a=0;
19. do {
20. q=v/u;
22. t=c;
23. c=a-q\*c;
24. a=t;
26. t=u;
27. u=v-q\*u;
28. v=t;
29. } while (u!=0);
30. a=a%y;
31. if (a<0) a=y+a;
32. return a;
33. }
35. */\* return (a^b) mod m \*/*
36. int pow\_mod(int a,int b,int m)
37. {
38. int r,aa;
40. r=1;
41. aa=a;
42. while (1) {
43. if (b&1) r=mul\_mod(r,aa,m);
44. b=b>>1;
45. if (b == 0) **break**;
46. aa=mul\_mod(aa,aa,m);
47. }
48. return r;
49. }
51. */\* return true if n is prime \*/*
52. int is\_prime(int n)
53. {
54. int r,i;
55. if ((n % 2) == 0) return 0;
57. r=(int)(sqrt(n));
58. for(i=3;i<=r;i+=2) if ((n % i) == 0) return 0;
59. return 1;
60. }
62. */\* return the prime number immediatly after n \*/*
63. int next\_prime(int n)
64. {
65. do {
66. n++;
67. } while (!is\_prime(n));
68. return n;
69. }
71. int main(int argc,char \*argv[])
72. {
73. int av,a,vmax,N,n,num,den,k,kq,kq2,t,v,s,i;
74. double sum;
76. if (argc<2 || (n=atoi(argv[1])) <= 0) {
77. printf("This program computes the n'th decimal digit of **\\**pi**\n**"
78. "usage: pi n , where n is the digit you want**\n**"
79. );
80. exit(1);
81. }
83. N=(int)((n+20)\*log(10)/log(2));
85. sum=0;
87. for(a=3;a<=(2\*N);a=next\_prime(a)) {
89. vmax=(int)(log(2\*N)/log(a));
90. av=1;
91. for(i=0;i<vmax;i++) av=av\*a;
93. s=0;
94. num=1;
95. den=1;
96. v=0;
97. kq=1;
98. kq2=1;
100. for(k=1;k<=N;k++) {
102. t=k;
103. if (kq >= a) {
104. do {
105. t=t/a;
106. v--;
107. } while ((t % a) == 0);
108. kq=0;
109. }
110. kq++;
111. num=mul\_mod(num,t,av);
113. t=(2\*k-1);
114. if (kq2 >= a) {
115. if (kq2 == a) {
116. do {
117. t=t/a;
118. v++;
119. } while ((t % a) == 0);
120. }
121. kq2-=a;
122. }
123. den=mul\_mod(den,t,av);
124. kq2+=2;
126. if (v > 0) {
127. t=inv\_mod(den,av);
128. t=mul\_mod(t,num,av);
129. t=mul\_mod(t,k,av);
130. for(i=v;i<vmax;i++) t=mul\_mod(t,a,av);
131. s+=t;
132. if (s>=av) s-=av;
133. }
135. }
137. t=pow\_mod(10,n-1,av);
138. s=mul\_mod(s,t,av);
139. sum=fmod(sum+(double) s/ (double) av,1.0);
140. }
141. printf("Decimal digits of pi at position %d: %09d**\n**",n,(int)(sum\*1e9));
142. return 0;
143. }

# Código Fonte Ruby Destrossator

1. *#!/usr/bin/ruby*
2. **require** 'date'
3. **require** 'thread\_pool'
5. **class** Destrossator
6. attr\_accessor **:keys**, **:keys\_to\_generate**, **:data\_rows\_per\_key**
7. **def** initialize**(**params = **{})**
8. **@keys\_to\_generate** = 1
9. **@data\_rows\_per\_key** = 1e6
10. **end**
12. **def** make\_keys**(**quantity = **@keys\_to\_generate)**
13. **@keys** = **[]**
14. **for** i **in** 1..quantity **do**
15. **@keys** **<<** **{** **:id** **=>** **rand(**999999**)**, **:sum** **=>** 0 **}**
16. **end**
17. **@keys**.sort! **{** **|**a, b**|** a**[:id]** **<=>** b**[:id]** **}**
18. **end**
20. **def** make\_data**(**rows\_per\_key = **@data\_rows\_per\_key)**
21. *#total\_iterations = @keys.length \* rows\_per\_key*
22. total\_iterations = rows\_per\_key
23. **@summary** = **File**.**open(**'dummy\_data.summary', 'w'**)**
24. **@dummy** = **File**.**open(**'dummy\_data', 'w'**)**
25. started\_on = **Time**.now
26. **@summary**.write "Started: #{started\_on.to\_s}\n";
27. pool = ThreadPool.new**(**10**)** *# up to 10 threads*
29. **for** i **in** 0..@keys\_to\_generate **do**
30. pool.process **do**
31. **for** ii **in** 0..@data\_rows\_per\_key **do**
32. index = **rand(**@keys\_to\_generate**)**
33. key = **@keys[**index**]**
34. val = **rand(**100**)**
35. key**[:sum]** **+**= val
36. **@dummy**.write "#{key[:id]},#{val}\n"
37. **end**
38. **end**
39. **end**
41. **@dummy**.close
42. **@keys**.each **do** **|**item**|**
43. **@summary**.write "#{item[:id]}   #{item[:sum]}\n"
44. **end**
46. ended\_on = **Time**.now;
47. **@summary**.write "Finished: #{ended\_on.to\_s}\n";
48. **@summary**.write "Elapsed : #{date\_diff(started\_on, ended\_on)}\n";
49. **@summary**.close
51. **end**
53. **def** date\_diff**(**started\_on, ended\_on**)**
54. difference = ended\_on **-** started\_on
55. seconds    =  difference **%** 60
56. difference = **(**difference **-** seconds**)** **/** 60
57. minutes    =  difference **%** 60
58. difference = **(**difference **-** minutes**)** **/** 60
59. hours      =  difference **%** 24
60. difference = **(**difference **-** hours**)**   **/** 24
61. days       =  difference **%** 7
62. weeks      = **(**difference **-** days**)**    **/**  7
64. "#{weeks} weeks, #{days} days, #{hours} hours, #{minutes} minutes, #{seconds} seconds"
65. **end**
67. **def** dump\_keys
68. **@keys**.each **do** **|**v**|**
69. **puts** "Id: #{v[:id]} Sum: #{v[:sum]}"
70. **end**
71. **end**
73. **end**
75. d = Destrossator.new
76. d.keys\_to\_generate = 1e3
77. d.data\_rows\_per\_key = 1e6
78. started = **Time**.now
79. d.make\_keys
80. d.make\_data
81. finished = **Time**.now
82. **puts**
83. **puts** "Finito (took #{d.date\_diff(started, finished)})"

86. *#!/usr/bin/ruby*
87. **require** 'thread'
88. **class** ThreadPool
89. **class** Worker
90. **def** initialize
91. **@mutex** = **Mutex**.new
92. **@thread** = **Thread**.new **do**
93. **while** **true**
94. **sleep** 0.001
95. block = get\_block
96. **if** block
97. block.call
98. reset\_block
99. **end**
100. **end**
101. **end**
102. **end**
104. **def** get\_block
105. **@mutex**.synchronize **{**@block**}**
106. **end**
108. **def** set\_block**(**block**)**
109. **@mutex**.synchronize **do**
110. **raise** **RuntimeError**, "Thread already busy." **if** **@block**
111. **@block** = block
112. **end**
113. **end**
115. **def** reset\_block
116. **@mutex**.synchronize **{**@block = **nil}**
117. **end**
119. **def** busy?
120. **@mutex**.synchronize **{**!@block.**nil**?**}**
121. **end**
122. **end**
124. attr\_accessor **:max\_size**
125. attr\_reader **:workers**
127. **def** initialize**(**max\_size = 10**)**
128. **@max\_size** = max\_size
129. **@workers** = **[]**
130. **@mutex** = **Mutex**.new
131. **end**
133. **def** size
134. **@mutex**.synchronize **{**@workers.size**}**
135. **end**
137. **def** busy?
138. **@mutex**.synchronize **{**@workers.any? **{|**w**|** w.busy?**}}**
139. **end**
141. **def** join
142. **sleep** 0.01 **while** busy?
143. **end**
145. **def** process**(&**block**)**
146. wait\_for\_worker.set\_block**(**block**)**
147. **end**
149. **def** wait\_for\_worker
150. **while** **true**
151. worker = find\_available\_worker
152. **return** worker **if** worker
153. **sleep** 0.01
154. **end**
155. **end**
157. **def** find\_available\_worker
158. **@mutex**.synchronize **{**free\_worker **||** create\_worker**}**
159. **end**
161. **def** free\_worker
162. **@workers**.each **{|**w**|** **return** w **unless** w.busy?**}**; **nil**
163. **end**
165. **def** create\_worker
166. **return** **nil** **if** **@workers**.size **>**= **@max\_size**
167. worker = Worker.new
168. **@workers** **<<** worker
169. worker
170. **end**
171. **end**

# Códigos de Algoritimos Genéricos C#

1. **using** System;
2. **using** System.Collections;
4. **namespace** ConsoleBenchBasic
5. {
6. **public** **class** TestClass
7. {
8. **static** **long** \_ticks;
10. **public** **static** **void** InitTicks()
11. {
12. \_ticks = DateTime.Now.Ticks;
13. }
15. **public** **static** **void** ShowTime(**string** str)
16. {
17. **long** newTicks = DateTime.Now.Ticks;
19. **double** ms;
20. ms = (newTicks - \_ticks) / TimeSpan.TicksPerMillisecond;
22. Console.WriteLine("{0}{1} ms", str.PadRight(35, '.'), ms);
23. \_ticks = DateTime.Now.Ticks;
24. }
26. */// <summary>*
27. */// 1 million strings são inseridos em uma  ArrayList.*
28. */// </summary>*
29. **public** **static** **void** Test1()
30. {
31. var a = new ArrayList();
32. **for** (**int** i = 0; i < 1000000; i++)
33. {
34. **string** str = i.ToString();
35. a.Add(str);
36. }
38. ShowTime("ArrayList strings teste");
39. var strBuilder = new System.Text.StringBuilder();
40. var rnd = new Random();
41. **foreach** (**object** strObj **in** a)
42. {
43. strBuilder.Append(strObj **as** **string**);
44. strBuilder.Append(rnd.Next().ToString());
45. }
47. **string** s = strBuilder.ToString();
49. **if** (s == "Dsadasdasdsa")
50. **throw** (new Exception("no way..."));
52. ShowTime("StringBuilder teste");
53. }

56. */// <summary>*
57. */// Um inteiro é adicionado a um double 1 bilhão de vezes.*
58. */// </summary>*
59. **public** **static** **void** Test2()
60. {
61. **int** i;
62. **double** d = 0.0;
63. **for** (i = 0; i < 1000000000; i++)
64. d += i;
66. **double** d2 = d / 2;
67. ShowTime("Adição de Integer & Floating");
68. }
70. */// <summary>*
71. */// 100.000 exceptions são jogadas.*
72. */// </summary>*
73. **public** **static** **void** Test3()
74. {
75. **int** i;
76. **int** sum = 0;
78. **for** (i = 0; i < 100000; i++)
79. **try**
80. {
81. **if** (i is **int**)
82. **throw** (new Exception("o inteiro  i é um inteiro.. oh no..."));
83. }
84. **catch** (Exception ex)
85. {
86. **if** (ex is NotImplementedException)
87. Console.Write("Quem é responsável por isso ?");
88. }
90. **int** res = sum / 2;
91. **if** (res == -321)
92. **throw** (new Exception("Isto é estranho..."));
94. ShowTime("Exception teste");
95. }

98. **public** **static** **void** Recursive(**string** str)
99. {
100. **string** name = System.Reflection.MethodBase.GetCurrentMethod().Name;
102. **if** (str.Substring(0, name.Length) == name)
103. {
104. **int** i = Convert.ToInt32(str.Substring(name.Length));
105. **if** (i < 1000)
106. Recursive(name + (i + 1));
107. }
109. }
110. */// <summary>*
111. */// Função recursiva que chama  GetCurrentMethod() é chamada por 1000 vezes.*
112. */// </summary>*
113. **public** **static** **void** Test4()
114. {
115. **for** (**int** i = 0; i < 1000; i++)
116. Recursive("Recursive0");
118. ShowTime("Reflection e recursão");
119. }
121. */// <summary>*
122. */// Programa que executa todos os testes.*
123. */// </summary>*
124. **public** **static** **void** Main()
125. {
126. InitTicks();
127. Test1();
128. Test2();
129. Test3();
130. Test4();
131. Console.WriteLine(  "Teste acabou");
132. Console.ReadKey();
133. }
134. }
135. }

# Código Fonte Acesso Banco de Dados C# - Single Thread

1. **using** System;
3. **namespace** G6DataLib
4. {
5. **public** **class** Record
6. {
7. **private** **static**  **long** serialVersionUID = 7098880197177237832L;
9. **private** **static**  **string** largeString = generateLargeString();
11. **private** **long** \_id;
12. **public** **long** Id
13. {
14. get { **return** \_id; }
15. set { \_id = value; }
16. }
18. **private** **string** \_name;
19. **public**  **string** Name
20. {
21. get { **return** \_name; }
22. set { \_name = value; }
23. }
25. **private** **string** \_string1;
26. **public** **string** String1
27. {
28. get { **return** \_string1; }
29. set { \_string1 = value; }
30. }
32. **private** **decimal** \_bigDecimal1;
33. **public** **decimal** BigDecimal1
34. {
35. get { **return** \_bigDecimal1; }
36. set { \_bigDecimal1 = value; }
37. }
39. **private** **decimal** \_bigDecimal2;
40. **public** **decimal** BigDecimal2
41. {
42. get { **return** \_bigDecimal2; }
43. set { \_bigDecimal2 = value; }
44. }
46. **private** DateTime \_date1;
47. **public** DateTime Date1
48. {
49. get { **return** \_date1; }
50. set { \_date1 = value; }
51. }
53. **private** DateTime \_date2;
54. **public** DateTime Date2
55. {
56. get { **return** \_date2; }
57. set { \_date2 = value; }
58. }

61. **private** **static** **string** generateLargeString()
62. {
64. **return** "XPTO";
65. }
67. **private** **static** **readonly** Random RANDOM = new Random();
69. **public** Record(**long** id) : **this**(id, RANDOM)
70. {
72. }
74. **public** Record(**long** id, **String** name, **String** string1, **Decimal** bigDecimal1, **Decimal** bigDecimal2, DateTime date1, DateTime date2)
75. {
76. \_id = id;
77. \_name = name;
78. \_string1 = string1;
79. \_bigDecimal1 = bigDecimal1;
80. \_bigDecimal2 = bigDecimal2;
81. \_date1 = date1;
82. \_date2 = date2;
83. }
85. **public** Record (**long** id, Random random) : **this**( id,"NAME" + (id % 10000), (id % 10000) == 0 ?largeString + id : **null**,
86. new **Decimal**(random.Next()),new **Decimal**(random.Next()),
87. new DateTime(random.Next()), new DateTime(random.Next())
88. )
89. {
91. }
93. *//public Record(long id, Random random)*
94. *//    : this(id, "NAME" + (id % 10000), (id % 10000) == 0 ? largeString + id : null,*
95. *//        new Decimal(random.Next()), new Decimal(random.Next()),*
96. *//        DateTime.Now, DateTime.Now*
97. *//        )*
98. *//{*
100. *//}*
102. **public** **int** hashCode()
103. {
104. **return** (**int**)(\_id
105. + \_name.GetHashCode()
106. + ("" + \_string1).GetHashCode()
107. + \_bigDecimal1.GetHashCode()
108. + \_bigDecimal2.GetHashCode()
109. + \_date1.GetHashCode()
110. + \_date2.GetHashCode()
111. );
112. }
114. }
115. }

118. **using** System;
120. **namespace** G6DataLib
121. {
122. **public** **interface** IScalabilityTestSubject
123. {
124. **string** Name();
125. **void** ReplaceAllRecords(**int** records);
126. **Object** CreateTestConnection();
127. **void** ReportResourcesUsed(**string** s);
128. }
129. }

132. **using** System;
133. **using** System.Data.SqlClient;
135. **namespace** ConsoleG6DataTest
136. {
137. **public** **class** ConectionTest
138. {
139. **private** **string** nomeServidorBancoDeDados;
140. **private** **string** nomeBancoDeDados;
141. **private** **string** usuarioBancoDeDados;
142. **private** **string** senhaBancoDeDados;
143. **public**  SqlConnection Connection;

146. **public** **bool** conectar(**bool** Amd)
147. {
148. **if** (Amd)
149. AmdDataBaseConfig();
150. **else**
151. z10DataBaseConfig();

154. **string** conexao = "user id=" + usuarioBancoDeDados + ";" + "password=" + senhaBancoDeDados +";server=" + nomeServidorBancoDeDados + ";" + "database=" + nomeBancoDeDados + "; " + "connection timeout=30";
155. Console.WriteLine(conexao);
156. Connection = new SqlConnection(conexao);
157. **try**
158. {
159. Connection.Open();
160. **return** **true**;
161. }
162. **catch** (Exception e)
163. {
164. Console.WriteLine(e.ToString());
165. }
166. **return** **false**;
167. }
169. **private** **void** AmdDataBaseConfig()
170. {
172. *// nomeServidorBancoDeDados = "g6srv2"; //(string)key.GetValue("SQLServer");*
173. nomeServidorBancoDeDados = "g6srv"; *//(string)key.GetValue("SQLServer");*
174. *//nomeBancoDeDados = "TesteMono"; // (string)key.GetValue("Database");*
175. nomeBancoDeDados = "TesteBinhara"; *// (string)key.GetValue("Database");*
176. usuarioBancoDeDados = "sa"; *//(string)key.GetValue("UserName");*
177. senhaBancoDeDados = ""; *// (string)key.GetValue("Password");*
179. }


183. **private** **void** z10DataBaseConfig()
184. {
186. nomeServidorBancoDeDados = "10.21.21.16"; *//(string)key.GetValue("SQLServer");*
187. nomeBancoDeDados = "TesteBinhara"; *// (string)key.GetValue("Database");*
188. usuarioBancoDeDados = "usuarioISS"; *//(string)key.GetValue("UserName");*
189. senhaBancoDeDados = "u$uarioI$$"; *// (string)key.GetValue("Password");*
191. }
192. }
193. }

196. **using** System;
197. **using** System.Data.SqlClient;
198. **using** G6DataLib;
200. **namespace** ConsoleG6DataTest
201. {

204. **public** **class** Program
205. {
206. **static** **long** \_ticks;
207. **private** **static** Record[] \_records;
208. **private** **static** **int** \_size;
209. **private** **static** ConectionTest \_conn;
211. **public** **static** **void** InitTicks()
212. {
213. \_ticks = DateTime.Now.Ticks;
214. }
216. **public** **static** **void** ShowTime(**string** str)
217. {
218. **long** newTicks = DateTime.Now.Ticks;
220. **double** ms;
221. *// ReSharper disable PossibleLossOfFraction*
222. ms = (newTicks - \_ticks) / TimeSpan.TicksPerMillisecond;
223. *// ReSharper restore PossibleLossOfFraction*
224. Console.WriteLine("{0}{1} ms", str.PadRight(35, '.'), ms);
225. \_ticks = DateTime.Now.Ticks;
226. }

229. **static** **void** Main(**string**[] args)
230. {
231. \_conn = new ConectionTest();
232. \_conn.conectar(args[0].Equals("linux"));
233. \_size = **int**.Parse(args[1]);
234. *//  \_conn.conectar(true);*
235. *//\_size = 2;*
237. \_records = new Record[\_size];
239. InitTicks();
240. TestCriaregistroEmMemoria();
241. TestSalvaRegistrosNoBanco();
242. TestBuscaDeRegistros();
243. TestUpdateRegistros();
244. TestRemocerRegistros();
245. Console.WriteLine( "Teste Finalizado");
246. Console.ReadKey();
248. }
250. **private** **static** **int** TestRemocerRegistros()
251. {
252. *//"delete from " + table() + " where ID=?");*
253. **int** executeNonQuery = 0;
254. **for** (**int** i = 0; i < \_size; i++)
255. {
256. **string** sql = "delete from " + table() + " where ID=" + \_records[i].Id;
257. SqlCommand cmd = new SqlCommand(sql, \_conn.Connection);
258. **try**
259. {
260. executeNonQuery = cmd.ExecuteNonQuery();
261. }
262. **catch** (Exception e)
263. {
264. Console.WriteLine("--> Error REMOVE : " + e.ToString());
265. }
267. }
268. ShowTime("-->> Remove " + \_size + " registros do Banco :");
269. **return** executeNonQuery;
270. }
272. **private** **static** **void** TestUpdateRegistros()
273. {
274. *//("update " + table() + " set NAME=?,STRING1=?,BIGDECIMAL1=?,BIGDECIMAL2=?,DATE1=?,DATE2=? where ID=?");*
275. **for** (**int** i = 0; i < \_size; i++)
276. {
277. Random r = new Random(i);
278. **int** key = r.Next(0, \_size);
280. **string** sql = "update " + table() + " set STRING1='UPDATE"+i+"' where ID=" +\_records[key].Id;
281. SqlCommand cmd = new SqlCommand(sql, \_conn.Connection);
283. **try**
284. {
285. cmd.ExecuteNonQuery();
286. }
287. **catch** (Exception e )
288. {
289. Console.WriteLine("-->> Error UPDATE : " + e.ToString());
290. }
291. }
292. ShowTime("-->> UPDATE " + \_size + " registros em memoria: ");
293. }
295. **private** **static** **void** TestBuscaDeRegistros()
296. {
297. *//"select ID,STRING1,BIGDECIMAL1,BIGDECIMAL2,DATE1,DATE2 from " + table() + " where NAME=?");*
299. **int** executeNonQuery;
300. **for** (**int** i = 0; i < \_size; i++)
301. {
302. Random r = new Random(i);
303. **int** key = r.Next(0, \_size);
304. **string** sql = "select ID,STRING1,BIGDECIMAL1,BIGDECIMAL2,DATE1,DATE2 from " + table() + " where NAME='" + \_records[key].Name+"'";
305. *//Console.WriteLine(  sql );*
306. SqlCommand cmd = new SqlCommand(sql, \_conn.Connection);
307. **try**
308. {
309. executeNonQuery = cmd.ExecuteNonQuery();
310. }
311. **catch** (Exception e)
312. {
314. Console.WriteLine("--> Error Busca: " + e.ToString());
315. }
316. }
317. ShowTime("-->> BUSCA " + \_size + " registros em memoria: ");
318. }
320. **private** **static** **int** TestSalvaRegistrosNoBanco()
321. {
322. **int** executeNonQuery=0;
323. **foreach** (var r **in** \_records)
324. {
325. *//      insertStatement = prepare("insert into " + table() + " (ID,NAME,STRING1,BIGDECIMAL1,BIGDECIMAL2,DATE1,DATE2) values(?,?,?,?,?,?,?)");*
326. **string** sql = "insert into " + table() + "  (ID,NAME,STRING1,BIGDECIMAL1,BIGDECIMAL2,DATE1,DATE2 ) values (" + r.Id +",'"+r.Name+ "','" + r.String1+ "'," + r.BigDecimal1 + "," + r.BigDecimal2 + ",'" + r.Date1 + "','" + r.Date2 + "')";
327. *//Console.WriteLine(  sql );*
328. SqlCommand cmd = new SqlCommand(sql, \_conn.Connection);
330. **try**
331. {
332. executeNonQuery = cmd.ExecuteNonQuery();
333. }
334. **catch** (Exception e)
335. {
337. Console.WriteLine( "-->Error insert:  "+  e.ToString() );
338. }
339. }
341. ShowTime("-->> inserido " + \_size + " registros no Banco: ");
342. **return** executeNonQuery;
343. }
345. **private** **static** **void** TestCriaregistroEmMemoria()
346. {
348. **for** (**int** i = 0; i < \_size; i++)
349. {
350. Random r = new Random(i);
351. \_records[i] = new Record(i,r);
352. }
354. ShowTime("-->> Cria " + \_size +" registros em memoria: ");
356. }
358. **protected** **static** **String** table()
359. {
360. **return** "QUERY\_TEST";
361. }
362. }
363. }

## Código Fonte Java de Escalabilidade

1. **package** org.prevayler.demos.scalability;
3. **import** org.prevayler.demos.scalability.prevayler.\*;
4. **import** org.prevayler.demos.scalability.jdbc.\*;
5. **import** org.prevayler.foundation.serialization.JavaSerializer;
7. **import** java.io.\*;
8. **import** java.util.\*;
10. **public** **class** Main {
12. **static** **private** **final** Properties properties = **new** Properties();

15. **static** **public** **void** main(String[] args) {


19. **try** {
20. out("Reading the properties file:**\n**" + propertiesFile().getAbsolutePath());
21. out("You can edit this file to configure the tests for the next run.**\n**");
23. properties.load(**new** FileInputStream(propertiesFile()));
25. **if** (isPrevaylerQueryChosen()) runPrevaylerQuery();
26. **if** (isPrevaylerTransactionChosen()) runPrevaylerTransaction();
27. **if** (isJdbcQueryChosen()) runJdbcQuery();
28. **if** (isJdbcTransactionChosen()) runJdbcTransaction();
30. out("**\n\n\n**For better results, edit the properties file:");
31. out(propertiesFile().getAbsolutePath());

34. } **catch** (Exception ex) {
35. ex.printStackTrace();
36. } **catch** (OutOfMemoryError err) {
37. ScalabilityTestRun.outOfMemory();
38. }
40. }

43. **static** **private** **void** runPrevaylerQuery() **throws** Exception {
44. **new** QueryTestRun(
45. **new** PrevaylerQuerySubject(),
46. numberOfObjects(),
47. prevaylerQueryThreadsMin(),
48. prevaylerQueryThreadsMax()
49. );
50. }
52. **static** **private** **void** runPrevaylerTransaction() **throws** Exception {
53. PrevaylerTransactionSubject subject = **new**PrevaylerTransactionSubject(prevaylerTransactionLogDirectory(), prevaylerJournalSerializer());
54. **new** TransactionTestRun(
55. subject,
56. numberOfObjects(),
57. prevaylerTransactionThreadsMin(),
58. prevaylerTransactionThreadsMax()
59. );
60. **if** (isPrevaylerTransactionConsistencyChecked()) {
61. out("Checking transaction log consistency.");
62. **if** (!subject.isConsistent()) **throw** **new** RuntimeException("Transaction log consistency check failed.");
63. out("Transaction log OK.**\n**");
64. }
65. }
67. **static** **private** **void** runJdbcQuery() {
68. **new** QueryTestRun(
69. **new** JDBCQuerySubject(jdbcDriverClassName(), jdbcConnectionURL(), jdbcUser(), jdbcPassword()),
70. numberOfObjects(),
71. jdbcQueryThreadsMin(),
72. jdbcQueryThreadsMax()
73. );
74. }
76. **static** **private** **void** runJdbcTransaction() {
77. **new** TransactionTestRun(
78. **new** JDBCTransactionSubject(jdbcDriverClassName(), jdbcConnectionURL(), jdbcUser(), jdbcPassword()),
79. numberOfObjects(),
80. jdbcTransactionThreadsMin(),
81. jdbcTransactionThreadsMax()
82. );
83. }

86. **static** **private** File propertiesFile() **throws** IOException {
87. File result = **new** File("ScalabilityTest.properties");
88. **if** (!result.exists()) {
89. out("Creating the properties file.");
90. createPropertiesFile(result);
91. }
92. **return** result;
93. }
95. **static** **private** **void** createPropertiesFile(File file) **throws** IOException {
96. PrintStream stream = **new** PrintStream(**new** FileOutputStream(file));
98. }

101. **static** **private** **int** numberOfObjects() {
102. String property = property("NumberOfObjects");
103. **if** ("ONE\_HUNDRED\_THOUSAND".equals(property)) **return**   100000;
104. **if** ("ONE\_MILLION"         .equals(property)) **return**  1000000;
105. **if** ("TEN\_MILLION"         .equals(property)) **return** 10000000;
106. **if** ("TWENTY\_MILLION"      .equals(property)) **return** 20000000;
107. **throw** **new** RuntimeException("NumberOfObjects property must be equal to ONE\_HUNDRED\_THOUSAND, ONE\_MILLION, TEN\_MILLION or TWENTY\_MILLION.");
108. }
110. **static** **private** **boolean** isPrevaylerQueryChosen() {
111. **return** booleanProperty("RunPrevaylerQueryTest");
112. }
114. **static** **private** **int** prevaylerQueryThreadsMin() {
115. **return** intProperty("PrevaylerQueryThreadsMinimum");
116. }
118. **static** **private** **int** prevaylerQueryThreadsMax() {
119. **return** intProperty("PrevaylerQueryThreadsMaximum");
120. }

123. **static** **private** **boolean** isPrevaylerTransactionChosen() {
124. **return** booleanProperty("RunPrevaylerTransactionTest");
125. }
127. **static** **private** **int** prevaylerTransactionThreadsMin() {
128. **return** intProperty("PrevaylerTransactionThreadsMinimum");
129. }
131. **static** **private** **int** prevaylerTransactionThreadsMax() {
132. **return** intProperty("PrevaylerTransactionThreadsMaximum");
133. }
135. **static** **private** **boolean** isPrevaylerTransactionConsistencyChecked() {
136. **return** booleanProperty("TransactionTestCheckConsistency");
137. }
139. **static** **private** String prevaylerTransactionLogDirectory() {
140. String result = property("TransactionLogDirectory");
141. out("**\n\n**Prevayler TransactionLog Directory: " + result);
142. **return** result;
143. }
145. **static** **private** String prevaylerJournalSerializer() {
146. String result = properties.getProperty("PrevaylerJournalSerializer");
147. **if** (result == **null**) result = JavaSerializer.**class**.getName();
148. out("**\n\n**Prevayler Journal Serializer: " + result);
149. **return** result;
150. }
152. **static** **private** **boolean** isJdbcQueryChosen() {
153. **return** booleanProperty("RunJdbcQueryTest");
154. }
156. **static** **private** **int** jdbcQueryThreadsMin() {
157. **return** intProperty("JdbcQueryThreadsMinimum");
158. }
160. **static** **private** **int** jdbcQueryThreadsMax() {
161. **return** intProperty("JdbcQueryThreadsMaximum");
162. }

165. **static** **private** **boolean** isJdbcTransactionChosen() {
166. **return** booleanProperty("RunJdbcTransactionTest");
167. }
169. **static** **private** **int** jdbcTransactionThreadsMin() {
170. **return** intProperty("JdbcTransactionThreadsMinimum");
171. }
173. **static** **private** **int** jdbcTransactionThreadsMax() {
174. **return** intProperty("JdbcTransactionThreadsMaximum");
175. }

178. **static** **private** String jdbcDriverClassName() {
179. **return** property("JdbcDriverClassName");
180. }
182. **static** **private** String jdbcConnectionURL() {
183. **return** property("JdbcConnectionURL");
184. }
186. **static** **private** String jdbcUser() {
187. **return** property("JdbcUser");
188. }
190. **static** **private** String jdbcPassword() {
191. **return** property("JdbcPassword");
192. }

195. **static** **private** String property(String name) {
196. String result = properties.getProperty(name);
197. **if** (result == **null**) **throw** **new** RuntimeException("Property " + name + " not found.");
198. **return** result;
199. }
201. **static** **private** **int** intProperty(String name) {
202. **try** {
203. **return** Integer.valueOf(property(name)).intValue();
204. } **catch** (NumberFormatException nfx) {
205. out("NumberFormatException reading property " + name);
206. **throw** nfx;
207. }
208. }
210. **static** **private** **boolean** booleanProperty(String name) {
211. **boolean** result = "yes".equalsIgnoreCase(property(name));
212. **if** (result) **return** **true**;
213. out("**\n\n\n**" + name + " property is set to " + property(name) + ".");
214. out("This test will be skipped (see properties file).");
215. **return** **false**;
216. }

219. **static** **private** **void** out(Object message) {
220. System.out.println(message);
221. }
222. }

225. **package** org.prevayler.demos.scalability;
227. **import** java.util.List;
229. **public** **interface** QueryConnection {
231. ***/\*\* Returns the List of all Record with the given name.***
232. ***\*/***
233. **public** List queryByName(String name);
234. }
235. **package** org.prevayler.demos.scalability;
237. **public** **class** QueryTestRun **extends** ScalabilityTestRun {
239. **public** QueryTestRun(ScalabilityTestSubject subject, **int** numberOfObjects, **int** minThreads, **int**maxThreads) {
240. **super**(subject, numberOfObjects, minThreads, maxThreads);
241. }

244. **protected** String name() {
245. **return** "Query Test";
246. }

249. **protected** **void** executeOperation(Object connection, **long** operationSequence) {
251. ((QueryConnection)connection).queryByName("NAME" + (operationSequence % 10000));
253. }
254. }

257. **package** org.prevayler.demos.scalability;
259. **import** java.io.Serializable;
260. **import** java.util.\*;
261. **import** java.math.BigDecimal;
263. **public** **class** Record **implements** Serializable {
265. **private** **static** **final** **long** serialVersionUID = 7098880197177237832L;
267. **private** **static** **final** String largeString = generateLargeString();
269. **private** **final** **long** id;
270. **private** **final** String name;
271. **private** **final** String string1;
272. **private** **final** BigDecimal bigDecimal1;
273. **private** **final** BigDecimal bigDecimal2;
274. **private** **final** **long** date1;
275. **private** **final** **long** date2;
277. **private** **static** **final** Random RANDOM = **new** Random();

280. **public** Record(**long** id) {
281. **this**(id, RANDOM);
282. }

285. **public** Record(**long** id, Random random) {
286. **this**(
287. id,
288. "NAME" + (id % 10000),
289. (id % 10000) == 0 ? largeString + id : **null**,
290. **new** BigDecimal(random.nextInt()),
291. **new** BigDecimal(random.nextInt()),
292. **new** Date(random.nextInt(10000000)),
293. **new** Date(random.nextInt(10000000))
294. );
295. }
297. **public** Record(**long** id, String name, String string1, BigDecimal bigDecimal1, BigDecimalbigDecimal2, Date date1, Date date2) {
298. **this**.id = id;
299. **this**.name = name;
300. **this**.string1 = string1;
301. **this**.bigDecimal1 = bigDecimal1;
302. **this**.bigDecimal2 = bigDecimal2;
303. **this**.date1 = date1.getTime();
304. **this**.date2 = date2.getTime();
305. }
307. **public** **long** getId() {
308. **return** id;
309. }
311. **public** String getName() {
312. **return** name;
313. }
315. **public** String getString1() {
316. **return** string1;
317. }
319. **public** BigDecimal getBigDecimal1() {
320. **return** bigDecimal1;
321. }
323. **public** BigDecimal getBigDecimal2() {
324. **return** bigDecimal2;
325. }
327. **public** Date getDate1() {
328. **return** **new** Date(date1);
329. }
331. **public** Date getDate2() {
332. **return** **new** Date(date2);
333. }
335. **public** **int** hashCode() {
336. **return** (**int**)(id
337. + name.hashCode()
338. + ("" + string1).hashCode()
339. + bigDecimal1.hashCode()
340. + bigDecimal2.hashCode()
341. + date1
342. + date2
343. );
344. }

347. **static** **private** String generateLargeString() {
348. **char**[] chars = **new** **char**[980];
349. Arrays.fill(chars, 'A');
350. **return** **new** String(chars);
351. }
353. }
355. **package** org.prevayler.demos.scalability;
357. **import** java.io.Serializable;
358. **import** java.util.Random;
360. ***/\*\* Generates Record objects with ids from 0 to numberOfRecords - 1.***
361. ***\*/***
362. **public** **class** RecordIterator **implements** Serializable {
364. **private** **static** **final** **long** serialVersionUID = 4278565982042966467L;
365. **private** **int** nextRecordId = 0;
366. **private** **final** **int** numberOfRecords;
367. **private** **final** Random \_random = **new** Random(0);

370. **public** RecordIterator(**int** numberOfRecords) {
371. **this**.numberOfRecords = numberOfRecords;
372. }
374. **public** **boolean** hasNext() {
375. **return** nextRecordId < numberOfRecords;
376. }
378. **public** Record next() {
379. indicateProgress();
380. **return** **new** Record(nextRecordId++, \_random);
381. }
383. **private** **void** indicateProgress() {
384. **if** (nextRecordId == 0) {
385. out("Creating " + numberOfRecords + " objects...");
386. **return**;
387. }
388. **if** (nextRecordId % 100000 == 0) out("" + nextRecordId + "...");
389. }
391. **static** **private** **void** out(Object message) {
392. System.out.println(message);
393. }
394. }

397. **package** org.prevayler.demos.scalability;
399. **import** org.prevayler.foundation.Cool;
400. **import** org.prevayler.foundation.StopWatch;
402. **import** java.text.DecimalFormat;
403. **import** java.util.LinkedList;
404. **import** java.util.List;
406. ***/\*\* Represents a single run of a scalability test. To understand the implementation of this class, you must be familiar with Prevayler's Scalability Test (run org.prevayler.test.scalability.ScalabilityTest).***
407. ***\*/***
408. **abstract** **class** ScalabilityTestRun {
410. **static** **private** **final** **long** ROUND\_DURATION\_MILLIS = 1000 \* 20;
412. **private** **final** ScalabilityTestSubject subject;
413. **protected** **final** **int** numberOfObjects;
415. **private** **double** bestRoundOperationsPerSecond;
416. **private** **int** bestRoundThreads;
418. **private** **final** List connectionCache = **new** LinkedList();
420. **private** **long** operationCount = 0;
421. **private** **long** lastOperation = 0;
422. **private** **boolean** isRoundFinished;
423. **private** **int** activeRoundThreads = 0;

426. ***/\*\* @return Example: "123.12 operations/second (12 threads)".***
427. ***\*/***
428. **public** String getResult() {
429. **return** toResultString(bestRoundOperationsPerSecond, bestRoundThreads);
430. }

433. **public** **double** getOperationsPerSecond() {
434. **return** bestRoundOperationsPerSecond;
435. }

438. **protected** ScalabilityTestRun(ScalabilityTestSubject subject, **int** numberOfObjects, **int**minThreads, **int** maxThreads) {
439. **if** (minThreads > maxThreads) **throw** **new** IllegalArgumentException("The minimum number of threads cannot be greater than the maximum number.");
440. **if** (minThreads < 1) **throw** **new** IllegalArgumentException("The minimum number of threads cannot be smaller than one.");
442. **this**.subject = subject;
443. **this**.numberOfObjects = numberOfObjects;
445. out("**\n\n**========= Running " + name() + " (" + (maxThreads - minThreads + 1) + " rounds). Subject: " + subject.name() + "...");
446. prepare();
448. out("Each round will take approx. " + ROUND\_DURATION\_MILLIS / 1000 + " seconds to run...");
449. performTest(minThreads, maxThreads);
450. out("**\n**----------- BEST ROUND: " + getResult());
451. }
453. **protected** **void** prepare() {
454. subject.replaceAllRecords(numberOfObjects);
455. System.gc();
456. }

459. ***/\*\* @return The name of the test to be executed. Example: "Prevayler Query Test".***
460. ***\*/***
461. **protected** **abstract** String name();

464. **private** **void** performTest(**int** minThreads, **int** maxThreads) {
466. **int** threads = minThreads;
467. **while** (threads <= maxThreads) {
468. **double** operationsPerSecond = performRound(threads);
470. **if** (operationsPerSecond > bestRoundOperationsPerSecond) {
471. bestRoundOperationsPerSecond = operationsPerSecond;
472. bestRoundThreads = threads;
473. }
475. threads++;
476. }
477. }

480. ***/\*\* @return The number of operations the test managed to execute per second with the given number of threads.***
481. ***\*/***
482. **private** **double** performRound(**int** threads) {
483. **long** initialOperationCount = operationCount;
484. StopWatch stopWatch = StopWatch.start();
486. startThreads(threads);
487. sleep();
488. stopThreads();
490. **double** secondsEllapsed = stopWatch.secondsEllapsed();
491. **double** operationsPerSecond = (operationCount - initialOperationCount) / secondsEllapsed;
493. out("**\n**Memory used: " + Runtime.getRuntime().totalMemory());
494. subject.reportResourcesUsed(System.out);
495. out("Seconds ellapsed: " + secondsEllapsed);
496. out("--------- Round Result: " + toResultString(operationsPerSecond, threads));
498. **return** operationsPerSecond;
499. }

502. **private** **void** startThreads(**int** threads) {
503. isRoundFinished = **false**;
505. **int** i = 1;
506. **while**(i <= threads) {
507. startThread(lastOperation + i, threads);
508. i++;
509. }
510. }

513. **private** **void** startThread(**final** **long** startingOperation, **final** **int** operationIncrement) {
514. (**new** Thread() {
515. **public** **void** run() {
516. **try** {
517. Object connection = acquireConnection();
519. **long** operation = startingOperation;
520. **while** (!isRoundFinished) {
521. executeOperation(connection, operation);
522. operation += operationIncrement;
523. }
525. **synchronized** (connectionCache) {
526. connectionCache.add(connection);
527. operationCount += (operation - startingOperation) /operationIncrement;
528. **if** (lastOperation < operation) lastOperation =operation;
529. activeRoundThreads--;
530. }
532. } **catch** (OutOfMemoryError err) {
533. outOfMemory();
534. }
535. }
536. }).start();
538. activeRoundThreads++;
539. }

542. **protected** **abstract** **void** executeOperation(Object connection, **long** operation);

545. **private** Object acquireConnection() {
546. **synchronized** (connectionCache) {
547. **return** connectionCache.isEmpty()
548. ? subject.createTestConnection()
549. : connectionCache.remove(0);
550. }
551. }

554. **private** **void** stopThreads() {
555. isRoundFinished = **true**;
556. **while** (activeRoundThreads != 0) {
557. Thread.yield();
558. }
559. }

562. **static** **private** String toResultString(**double** operationsPerSecond, **int** threads) {
563. String operations = **new** DecimalFormat("0.00").format(operationsPerSecond);
564. **return** "" + operations + " operations/second (" + threads + " threads)";
565. }
567. **static** **void** outOfMemory() {
568. System.gc();
569. out(
570. "**\n\n**OutOfMemoryError.**\n**" +
571. "===========================================================**\n**" +
572. "The VM must be started with a sufficient maximum heap size.**\n**" +
573. "Example for Linux and Windows:  java -Xmx512000000 ...**\n\n**"
574. );
575. }
577. **static** **private** **void** sleep() {
578. Cool.sleep(ROUND\_DURATION\_MILLIS);
579. }

582. **static** **private** **void** out(Object obj) {
583. System.out.println(obj);
584. }
585. }

588. **package** org.prevayler.demos.scalability;
590. **public** **interface** TransactionConnection {
592. **public** **void** performTransaction(Record recordToInsert, Record recordToUpdate, **long** idToDelete);
594. }
595. **package** org.prevayler.demos.scalability.jdbc;
597. **import** org.prevayler.demos.scalability.\*;
598. **import** java.sql.\*;
599. **import** java.util.\*;

602. *// Contributions by Stefan Ortmanns.*
603. **class** JDBCQueryConnection **extends** JDBCScalabilityConnection **implements** QueryConnection {
605. **private** **final** PreparedStatement selectStatement;

608. JDBCQueryConnection(Connection connection) {
609. **super**(connection);
611. selectStatement = prepare("select ID,STRING1,BIGDECIMAL1,BIGDECIMAL2,DATE1,DATE2 from "+ table() + " where NAME=?");
612. }

615. **protected** String table() {
616. **return** "QUERY\_TEST";
617. }

620. **public** List queryByName(String name) {
621. ArrayList list = **new** ArrayList();
622. **try** {
623. selectStatement.setString(1, name);
624. ResultSet resultSet = selectStatement.executeQuery();
626. **while** (resultSet.next()) {
627. list.add(**new** Record(resultSet.getLong(1), name, resultSet.getString(2), resultSet.getBigDecimal(3), resultSet.getBigDecimal(4), resultSet.getDate(5), resultSet.getDate(6)));
628. }
630. } **catch** (SQLException sqlex) {
631. dealWithSQLException(sqlex, "selecting record from " + table());
632. }
634. **return** list;
635. }
636. }
638. **package** org.prevayler.demos.scalability.jdbc;
640. **import** org.prevayler.demos.scalability.\*;
641. **import** java.sql.\*;

644. *// Contributions by Stefan Ortmanns.*
645. **abstract** **class** JDBCScalabilityConnection {
647. **protected** **final** Connection connection;
648. **private** **final** PreparedStatement insertStatement;
650. **protected** JDBCScalabilityConnection(Connection connection) {
651. **this**.connection = connection;
652. **try** {
653. connection.setAutoCommit(**false**);
654. } **catch** (SQLException sqlx) {
655. dealWithSQLException(sqlx, "setting AutoCommit to false");
656. }
658. insertStatement = prepare("insert into " + table() + " (ID,NAME,STRING1,BIGDECIMAL1,BIGDECIMAL2,DATE1,DATE2) values(?,?,?,?,?,?,?)");
659. }

662. **protected** **abstract** String table();

665. **protected** **void** insert(Record recordToInsert) {
666. **try** {
667. insertStatement.setLong(1,recordToInsert.getId());
668. insertStatement.setString(2,recordToInsert.getName());
669. insertStatement.setString(3,recordToInsert.getString1());
670. insertStatement.setBigDecimal(4,recordToInsert.getBigDecimal1());
671. insertStatement.setBigDecimal(5,recordToInsert.getBigDecimal2());
672. insertStatement.setDate(6,**new**java.sql.Date(recordToInsert.getDate1().getTime()));
673. insertStatement.setDate(7,**new**java.sql.Date(recordToInsert.getDate2().getTime()));
674. insertStatement.execute();
675. } **catch** (SQLException sqlx) {
676. dealWithSQLException(sqlx, "inserting record");
677. }
678. }

681. **protected** PreparedStatement prepare(String statement) {
682. **try** {
683. **return** connection.prepareStatement(statement);
684. } **catch** (SQLException sqlx) {
685. dealWithSQLException(sqlx, "preparing statement: " + statement);
686. **return** **null**;
687. }
688. }

691. **void** replaceAllRecords(**int** records) {
692. RecordIterator newRecords = **new** RecordIterator(records);
694. **try** {
695. connection.createStatement().execute("delete from " + table());
696. } **catch** (SQLException sqlx) {
697. dealWithSQLException(sqlx, "deleting all records from " + table());
698. }
700. **while** (newRecords.hasNext()) {
701. insert(newRecords.next());
702. }
704. **try** {
705. connection.commit();
706. } **catch** (SQLException sqlx) {
707. dealWithSQLException(sqlx, "commiting insertion of test records");
708. }
709. }

712. **static** **protected** **void** dealWithSQLException(SQLException sqlx, String duringOperation) {
713. sqlx.printStackTrace();
714. **throw** **new** RuntimeException("SQLException " + duringOperation + ".");
715. }
716. }

719. **package** org.prevayler.demos.scalability.jdbc;
721. **import** org.prevayler.demos.scalability.\*;
722. **import** java.sql.\*;
723. **import** java.io.PrintStream;

726. **abstract** **class** JDBCScalabilitySubject **implements** ScalabilityTestSubject {
728. **protected** **final** String connectionURL;
729. **protected** **final** String user;
730. **protected** **final** String password;
732. {System.gc();}

735. **protected** JDBCScalabilitySubject(String jdbcDriverClassName, String connectionURL, String user,String password) {
736. **try** {
737. **Class**.forName(jdbcDriverClassName);
738. } **catch** (Exception ex) {
739. ex.printStackTrace();
740. **throw** **new** RuntimeException("Exception loading JDBC driver class: " +jdbcDriverClassName);
741. }
743. **this**.connectionURL = connectionURL;
744. **this**.user = user;
745. **this**.password = password;
746. }
748. **public** String name() {
749. **return** "JDBC";
750. }
752. **public** **void** replaceAllRecords(**int** records) {
753. ((JDBCScalabilityConnection)createTestConnection()).replaceAllRecords(records);
754. }
756. **protected** Connection createConnection() {
757. **try** {
759. **return** DriverManager.getConnection(connectionURL, user, password);
761. } **catch** (SQLException sqlx) {
762. sqlx.printStackTrace();
763. **throw** **new** RuntimeException("Exception while trying to connect: " + sqlx);
764. }
765. }
767. **public** **void** reportResourcesUsed(PrintStream out) {
768. }
770. }


774. **package** org.prevayler.demos.scalability.jdbc;
776. **import** org.prevayler.demos.scalability.\*;
777. **import** java.sql.\*;

780. **class** JDBCTransactionConnection **extends** JDBCScalabilityConnection **implements** TransactionConnection {
782. **private** **final** PreparedStatement updateStatement;
783. **private** **final** PreparedStatement deleteStatement;

786. JDBCTransactionConnection(Connection connection) {
787. **super**(connection);
789. updateStatement = prepare("update "      + table() + " set NAME=?,STRING1=?,BIGDECIMAL1=?,BIGDECIMAL2=?,DATE1=?,DATE2=? where ID=?");
790. deleteStatement = prepare("delete from " + table() + " where ID=?");
791. }

794. **protected** String table() {
795. **return** "TRANSACTION\_TEST";
796. }

799. **public** **void** performTransaction(Record recordToInsert, Record recordToUpdate, **long** idToDelete) {
800. insert(recordToInsert);
801. update(recordToUpdate);
802. delete(idToDelete);
804. **try** {
805. connection.commit();
806. } **catch** (SQLException sqlx) {
807. dealWithSQLException(sqlx, "commiting transaction");
808. }
809. }

812. **private** **void** update(Record recordToUpdate) {
813. **try** {
814. updateStatement.setString(1,recordToUpdate.getName());
815. updateStatement.setString(2,recordToUpdate.getString1());
816. updateStatement.setBigDecimal(3,recordToUpdate.getBigDecimal1());
817. updateStatement.setBigDecimal(4,recordToUpdate.getBigDecimal2());
818. updateStatement.setDate(5,**new**java.sql.Date(recordToUpdate.getDate1().getTime()));
819. updateStatement.setDate(6,**new**java.sql.Date(recordToUpdate.getDate2().getTime()));
820. updateStatement.setLong(7,recordToUpdate.getId());  *//"...where ID=?"*
821. updateStatement.execute();
822. } **catch** (SQLException sqlx) {
823. dealWithSQLException(sqlx, "updating record");
824. }
825. }

828. **private** **void** delete(**long** idToDelete) {
829. **try** {
830. deleteStatement.setLong(1,idToDelete);
831. deleteStatement.execute();
832. } **catch** (SQLException sqlx) {
833. dealWithSQLException(sqlx, "deleting record");
834. }
835. }
836. }
838. **package** org.prevayler.demos.scalability.prevayler;
840. **import** java.util.Date;
842. **import** org.prevayler.Transaction;
843. **import** org.prevayler.demos.scalability.RecordIterator;
845. **class** AllRecordsReplacement **implements** Transaction {
847. **private** **static** **final** **long** serialVersionUID = 6283032417365727408L;
848. **private** **final** **int** \_records;
850. AllRecordsReplacement(**int** records) { \_records = records; }
852. **public** **void** executeOn(Object system, Date ignored) {
853. ((ScalabilitySystem)system).replaceAllRecords(**new** RecordIterator(\_records));
854. }
855. }
856. **package** org.prevayler.demos.scalability.prevayler;
858. **import** java.io.File;
859. **import** java.io.FileFilter;

862. **public** **class** PrevalenceTest {
864. **static** **public** **void** delete(String dir) {
865. delete(**new** File(dir));
866. }
868. **static** **private** **void** delete(File file) {
869. **if** (file.isDirectory()) deleteDirectoryContents(file);
870. **if** (!file.delete()) {
871. System.gc();
872. }
873. }
875. **static** **private** **void** deleteDirectoryContents(File directory) {
876. File[] files = directory.listFiles(**new** PrevalenceFilter());
877. **if** (files == **null**) **return**;
878. **for** (**int** i = 0; i < files.length; i++) delete(files[i]);
879. }
881. **static** **private** **class** PrevalenceFilter **implements** FileFilter {
882. **public** **boolean** accept(File file) {
883. **return** file.getName().endsWith("journal")
884. || file.getName().endsWith("snapshot")
885. || file.isDirectory();
886. }
887. }
889. }

892. package org.prevayler.demos.scalability.prevayler;
894. import org.prevayler.demos.scalability.\*;
895. import java.util.List;
897. class PrevaylerQueryConnection implements QueryConnection {
899. private final QuerySystem querySystem;

902. PrevaylerQueryConnection(QuerySystem querySystem) {
903. this.querySystem = querySystem;
904. }

907. public List queryByName(String name) {
908. return querySystem.queryByName(name);
909. }
910. }
912. **package** org.prevayler.demos.scalability.prevayler;
914. **import** java.io.File;
915. **import** java.io.PrintStream;
917. **import** org.prevayler.PrevaylerFactory;
918. *//import org.prevayler.implementation.PrevalenceTest;*
920. **public** **class** PrevaylerQuerySubject **extends** PrevaylerScalabilitySubject {
922. **static** **final** String PREVALENCE\_BASE = "QueryTest";
924. **public** PrevaylerQuerySubject() **throws** java.io.IOException, ClassNotFoundException {
925. **if** (**new** File(PREVALENCE\_BASE).exists()) PrevalenceTest.delete(PREVALENCE\_BASE);
926. PrevaylerFactory factory = **new** PrevaylerFactory();
927. factory.configurePrevalentSystem(**new** QuerySystem());
928. factory.configurePrevalenceDirectory(PREVALENCE\_BASE);
929. factory.configureTransactionFiltering(**false**);
930. prevayler = factory.create();
931. }

934. **public** Object createTestConnection() {
935. **return** **new** PrevaylerQueryConnection((QuerySystem)prevayler.prevalentSystem());
936. }
938. **public** **void** reportResourcesUsed(PrintStream out) {
939. }
941. }
942. **package** org.prevayler.demos.scalability.prevayler;
944. **import** java.io.File;
945. **import** java.io.PrintStream;
947. **import** org.prevayler.PrevaylerFactory;
948. *//import org.prevayler.implementation.PrevalenceTest;*
950. **public** **class** PrevaylerQuerySubject **extends** PrevaylerScalabilitySubject {
952. **static** **final** String PREVALENCE\_BASE = "QueryTest";
954. **public** PrevaylerQuerySubject() **throws** java.io.IOException, ClassNotFoundException {
955. **if** (**new** File(PREVALENCE\_BASE).exists()) PrevalenceTest.delete(PREVALENCE\_BASE);
956. PrevaylerFactory factory = **new** PrevaylerFactory();
957. factory.configurePrevalentSystem(**new** QuerySystem());
958. factory.configurePrevalenceDirectory(PREVALENCE\_BASE);
959. factory.configureTransactionFiltering(**false**);
960. prevayler = factory.create();
961. }

964. **public** Object createTestConnection() {
965. **return** **new** PrevaylerQueryConnection((QuerySystem)prevayler.prevalentSystem());
966. }
968. **public** **void** reportResourcesUsed(PrintStream out) {
969. }
971. }
972. **package** org.prevayler.demos.scalability.prevayler;
974. **import** java.util.ArrayList;
975. **import** java.util.Collections;
976. **import** java.util.HashMap;
977. **import** java.util.Iterator;
978. **import** java.util.List;
979. **import** java.util.Map;
981. **import** org.prevayler.demos.scalability.Record;
982. **import** org.prevayler.demos.scalability.RecordIterator;
984. **class** QuerySystem **implements** ScalabilitySystem {
986. **private** **static** **final** **long** serialVersionUID = -8181198939095700706L;
987. **private** Map recordsByName = **new** HashMap();

990. **public** List queryByName(String name) {
991. **return** (List)recordsByName.get(name);
992. }

995. **public** **void** replaceAllRecords(RecordIterator newRecords) {
996. recordsByName.clear();
998. **while** (newRecords.hasNext()) {
999. put(newRecords.next());
1000. }
1002. makeReadOnly();
1003. }

1006. **private** **void** put(Record newRecord) {
1007. List records = queryByName(newRecord.getName());
1008. **if** (records == **null**) {
1009. records = **new** ArrayList();
1010. recordsByName.put(newRecord.getName(), records);
1011. }
1013. records.add(newRecord);
1014. }

1017. ***/\*\* This is necessary so that the clients cannot alter the Lists they receive as query results.***
1018. ***\*/***
1019. **private** **void** makeReadOnly() {
1020. Iterator entries = recordsByName.entrySet().iterator();
1021. **while** (entries.hasNext()) {
1022. Map.Entry entry = (Map.Entry)entries.next();
1023. entry.setValue(Collections.unmodifiableList((List)entry.getValue()));
1024. }
1025. }
1026. }

Slides da Apresentação



