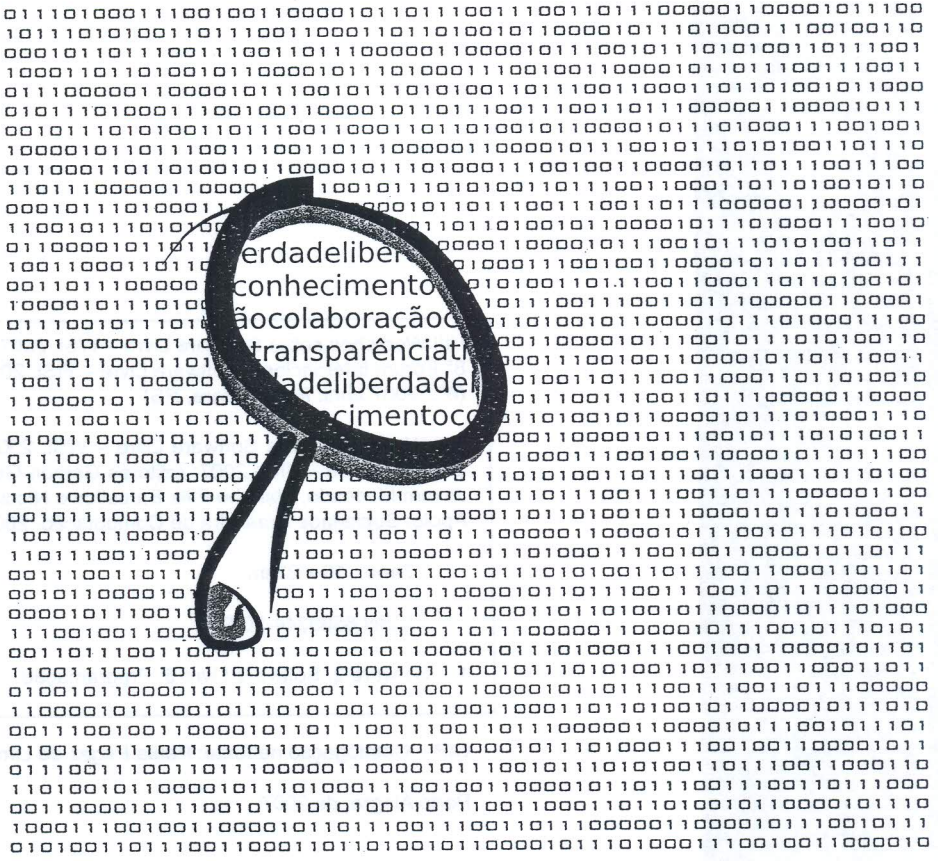
fisl8.0

8° Fórum Internacional Software Livre

A tecnologia que liberta



VIII Workshop sobre Software Livre

12, 13 e 14 de abril de 2007 - http://fisl.softwarelivre.org
Centro de Eventos FIERGS - Porto Alegre - RS - Brasil

Capa: Studio 11

Impressão e Acabamento:
Organizações Nova Prova Gráfica e Editora Ltda.

Tiragem:
4.000 exemplares

VIII Workshop sobre Software Livre – WSL 2007-04-09
8° Fórum Internacional Software Livre – FISL 2007
(8°.: 2007 abril: Porto Alegre, RS)

VIII Workshop sobre Software Livre – WSL 2007 / 8° Fórum Internacional Software Livre – FISL 2007; editores Gerson Geraldo Homrich Cavalheiro, Javam Machado, Marinho Pilla Barcellos e Nicolas Maillard. – Porto Alegre. Apoio: Sociedade Brasileira de Computação, 2007.

344p.; 16 x 23cm

ISBN 8589344-50-9

1. Sofware 2. Software Livre 3. Programação

CIP – Catalogação na fonte: Paula Pegas de Lima CRB 10/ 1229

Porto Alegre, de 2007

Melhorando a performance de aplicações científicas escritas na linguagem Python

Luiz Carlos Irber Júnior[1], Lúcio André de Castro Jorge[1]

[1]Embrapa Instrumentação Agropecuária (CNPDIA)
Rua XV de Novembro, 1452 – 13.560-970 – São Carlos – SP – Brasil

luiz.irber@gmail.com, lucio@cnpdia.embrapa.br

Abstract. *The utilization of the programming language Python in scientific areas is rising in recent years, but some researchers still have doubts about its usage because of its supposed low performance. This article shows some solutions adopted for the criation and design of a animal motion behavior analysis system, aiming to simplify the development and maintainability of the software using Python and, at the same time, keeping the system at a good performance.*

Resumo. *A utilização da linguagem de programação Python no meio científico vem crescendo nos últimos anos, mas alguns pesquisadores ainda restringem seu uso devido à suposta baixa performance da linguagem. Este artigo apresenta algumas soluções adotadas para a criação de um sistema para análise comportamental de animais em movimento, visando facilitar o desenvolvimento e a manutenção do software através do uso de Python e, ao mesmo tempo, manter o bom desempenho do sistema.*

1. Introdução

No escopo deste artigo serão discutidos detalhes da implementação de um sistema de análise comportamental de animais em movimento chamado SACAM, disponível em `http://bugbrother.sourceforge.net`. Este sistema foi desenvolvido no Laboratório de Imagem da Embrapa Instrumentação Agropecuária, em São Carlos, conforme requisição da Embrapa Recursos Genéticos e Biotecnologia, que o utiliza para automação do processo de análise da ação de semioquímicos, como feromônios, no comportamento de insetos. Foi escrito na linguagem de programação Python, uma linguagem interpretada e dinâmica. Linguagens de programação deste tipo costumam ter performance inferior a linguagens compiladas e estáticas, mas com a utilização de algumas técnicas as primeiras podem atingir performances muito próximas às segundas.

No caso deste sistema, apenas uma pequena parte demandou alta performance, já que a maior parte do sistema apenas realiza interação com o usuário, uma tarefa limitada não tanto pela performance da linguagem mas sim pela velocidade de com que o usuário interage com o sistema, que é muito inferior àquelas alcançadas pela linguagem. O sistema é interessante por ser um caso extremo, onde há uma necessidade constante de processamento por parte da CPU e dos dispositivos de E/S quando ocorre a captura de vídeo e detecção de movimento, e a partir dessa necessidade podem-se discutir técnicas para otimizar o desempenho, de forma que sistemas mais simples também possam utilizar essas técnicas.

2. Tecnologias livres utilizadas

2.1. Python

Python é uma linguagem de programação interpretada criada por Guido van Rossum no CWI, na Holanda, em 1990. Sua sistema de tipos é totalmente dinâmico e usa a estratégia de coletor de lixo para gerenciamento automático de memória. Hoje em dia Python é desenvolvido como um projeto de software livre e gerenciado pela organização sem fins lucrativos Python Software Foundation.

Como uma linguagem de programação multi-paradigmas, desenvolvedores de Python podem usar ferramentas da programação orientação a objetos, estruturada, funcional ou orientada a aspectos, e outros paradigmas podem ser suportados usando extensões. Apesar de isso poder levar a uma sintaxe confusa, desenvolvedores rejeitam sintaxes exuberantes em favor de uma mais simples e menos densa. De fato, transparência e clareza de escrita são altamente encorajadas, pois a comunidade tende a rejeitar código que é denso ou complicado. Isso não significa que algoritmos complexos não possam ser implementados, mas que os desenvolvedores devem mantê-los o mais claro e simples possível.

Um dos aspectos que colaboram para esse objetivo é que, diferente de C, blocos de código são delimitados pela indentação ao invés de chaves. Um acréscimo na indentação indica o início de um bloco após certas expressões (definição de função, por exemplo), e um recuo na indentação significa o fim do bloco atual. Apesar de muitos programadores pensarem que isso é uma restrição ao estilo de codificação, de fato isto aumenta consideravelmente a clareza do código fonte.

Outro aspecto importante é a extensibilidade. Novos módulos podem ser escritos facilmente em Python ou então em C ou C++. Python também pode ser usado como uma linguagem de "cola", juntando módulos existentes e aplicações que necessitem de uma interface programável.

Devido à sua modularidade, Python tem uma extensiva biblioteca padrão, cobrindo muitos módulos úteis como parsers de XML e HTML, suporte a threads e utilitários para distribuição de software que aumentam a produtividade do desenvolvedor. Além da biblioteca padrão existem muitos pacotes contendo módulos para aplicações de domínios específicos, como SciPy, uma biblioteca de rotinas científicas e numéricas, e BioPython, que provê ferramentas para o campo da biologia molecular computacional.

2.2. GTK+

GTK+ é uma biblioteca de widgets originalmente escrita para o X Window System, mas agora está disponível para outros sistemas, como o Windows. É implementada em C, mas seus criadores utilizaram a orientação a objeto como paradigma. Existem várias extensões para outras linguagens, como C++ (GTKmm), Java e Python (PyGTK), entre muitas outras.

A GTK+ é, na verdade, baseada em outras quatro bibliotecas, para facilitar a sua portabilidade. São elas:

- Glib, a biblioteca de baixo nível, que forma a base da GTK+ e do GNOME. Provê manipulação de estruturas de dados, encapsulamentos para portabilidade, e interfaces para funcionalidades de execução como um laço de eventos, threads, carregamento dinâmico e um sistema de objetos (GObject).

- Pango, uma biblioteca para layout e renderização de texto, com ênfase em internacionalização.
- ATK, biblioteca que provê um conjunto de interfaces para acessibilidade. Por suportar as interfaces da ATK, uma aplicação pode ser usada com ferramentas como leitores de telas, lentes de aumento ou métodos alternativos de entrada de dados.
- GDK, sigla para GTK+ Drawing Kit, é a biblioteca que disponibiliza as funções primitivas que lidam com tarefas específicas do sistema gráfico, como desenhar pixels ou redimensionar janelas.

2.3. GStreamer

GStreamer é um framework multimídia livre. Por prover uma API padrão e encapsular todas as ferramentas e bibliotecas dentro de plugins ela simplifica o desenvolvimento de aplicações baseadas em mídia, como editores de som e servidores de streaming, por exemplo. Para isso o GStreamer usa o conceito de *pipeline*, um grafo direcionado onde a mídia flui de uma entrada para uma saída. Dentro de uma pipeline existem *elements*, objetos que empacotam uma funcionalidade, como adquirir vídeo de um dispositivo de captura, rotacionar um vídeo, codificar em Ogg Vorbis um áudio, e assim por diante. Ligando esses elementos através de *pads*, que são as entradas e saídas de um elemento, o desenvolvedor pode criar uma pipeline completa que implementa operações complexas.

Uma das principais vantagens do GStreamer é que, contanto que você ache o plugin certo ou o escreva, as pipelines são extensíveis: um programa não precisa se restringir a capturar vídeo de uma placa de captura, quando pode capturar o mesmo vídeo de uma webcam, ou de um servidor de streaming, por exemplo, apenas mudando o elemento que provê a entrada da pipeline. Como pipelines podem ser definidas em formato XML, o programa sequer precisa ser recompilado, se uma linguagem compilada estiver sendo usada.

Como o GStreamer é escrito em C (utilizando GObject, a biblioteca da GTK+ que implementa orientação a objetos em C) ele é muito portável e, de fato, está disponível para sistemas operacionais como Linux, Solaris, MacOS X e Windows. Assim como a GTK+, existem camadas de compatibilidade para linguagens como Python, Ruby, C++ e todas as linguagens suportadas pelo Common Language Runtime do framework .NET.

2.4. Razões para a escolha das tecnologias utilizadas

A escolha de Python como linguagem de programação foi feita devido ao rápido ciclo de desenvolvimento e à facilidade de manutenção do código. O ciclo de desenvolvimento é acelerado devido ao fato de Python ser uma linguagem interpretada, ou seja, não é necessário recompilar o código a cada mudança.

A biblioteca gráfica GTK+ foi escolhida por estar disponível em vários sistemas operacionais (principalmente GNU/Linux e Windows, os dois sistemas alvo na primeira etapa). Existiam outras opções, como Qt ou WxWidgets, mas a GTK+ ofereceu as melhores ferramentas de desenvolvimento e uma maior comunidade de desenvolvedores.

Por fim, a opção pela biblioteca GStreamer decorreu de sua extensibilidade e performance. Historicamente não existiu um framework unificado para multimídia em sistemas operacionais de código aberto como o Quicktime no MacOS ou o DirectShow no Windows.

3. Decisões de design

Desde o início, a orientação a objetos foi o paradigma adotado. Desse modo, utilizou-se o conceitos de classes para fazer a modelagem do software. A classe principal, chamada Interface, é composta por agregação de objetos das classes Projeto, Gerenciador de Dispositivos e de outras, responsáveis pelas janelas de diálogo do sistema. Além disso, ela implementa a janela principal do programa e os métodos necessários para a interação com o usuário.

Para a implementação da classe Interface não é necessária uma grande performance, pois ela é limitadas pela velocidade com que o usuário interage com o sistema. E, de qualquer modo, como a biblioteca gráfica GTK+ está sendo utilizada, através de sua interface para Python (chamada PyGTK), essa performance é muito mais do que o suficiente, já que a GTK+ é implementada em C e apresenta uma performance muito boa.

A subclasse Projeto contém os atributos necessários para a definição de um projeto, além de ser composto por uma lista de experimentos. Cada experimento é um objeto da classe Experimento, que implementa os atributos necessários para o cálculo das estatísticas que o software gera.

Para o processamento dessas estatísticas seriam necessárias capacidades de manipulação de listas, já que os principais dados utilizados para a geração são pontos, que contém a posição e o tempo decorrido desde o início do experimento. Verificando a documentação do Python pode-se perceber que um grande esforço foi feito na otimização da performance da manipulação de listas, já que essas são um dos elementos básicos que a linguagem provê. Verificar o desempenho nos testes foi o suficiente para optar por manter essa funcionalidade em Python puro, sem uso de extensões ou outros módulos.

3.1. Gerenciador de Dispositivos

Para boa parte da classe Gerenciador de Dispositivos, o raciocínio aplicado à classe Interface é coerente. Esta classe implementa os métodos que fazem a interface com o GStreamer, além de janelas de diálogo para a configuração da entrada de vídeo.

O mesmo não pode ser dito do Processador de Vídeo, classe responsável pela detecção de movimento. Aqui, o fato de Python ser interpretada e de seu sistema de tipos ser dinâmico ofereceu bons argumentos para os detratores da linguagem, que afirmam que Python é muito lento para aplicações que demandem performance, como as científicas.

3.1.1. Algoritmo de detecção de movimentos

Uma das abordagens mais simples para detecção de movimento, e que foi adotada neste sistema, é a simples detecção de mudanças entre duas imagens obtidas seqüencialmente num vídeo. Sejam as imagens $f(x, y, t_i)$ e $f(x, y, t_j)$ obtidas nos tempos t_i e t_j, respectivamente, é feita a comparação das imagens pixel a pixel pela diferença dada pela seguinte equação:

$$d_{ij} = \begin{cases} 1 & \text{para } |f(x,y,t_i) - f(x,y,t_j)| > \theta \\ 0 & \text{caso contrário} \end{cases}$$

onde o θ é o limiar. O valor de $d_{ij}(x, y)$ é 1 para todo pixel (x, y) somente quando as diferenças entre as duas imagens naquele ponto for acima do valor estabelecido no limiar.

3.1.2. Melhorando a performance do Processador de Vídeo

O primeiro passo para melhorar a performance do processador de vídeo foi definir o limiar não apenas para cada pixel isolado, mas sim pelo valor médio de uma janela w, proporcional ao tamanho do inseto e à sua velocidade máxima. Deste modo, não é necessário aplicar o algoritmo à imagem inteira, o que já auxilia muito na performance.

O segundo passo foi procurar maneiras mais eficientes para manipular matrizes. Enquanto a manipulação de listas é otimizada, como descrito acima, a manipulação de matrizes em Python tem problemas sérios de desempenho, e não é recomendada [Ramachandran 2007]. A primeira tentativa foi a utilização da biblioteca NumPy [NumPy 2007], uma biblioteca desenvolvida de modo a ser básica para computação no meio científico, que implementa módulos para funções elementares de algebra linear, transformadas de Fourier, geradores de números aleatórios e, principalmente, um objeto vetor N-dimensional, com otimizações para melhora da performance.

Os testes realizados mostraram-se promissores, mas o vetor N-dimensional do NumPy ainda não apresentou a performance desejada. O próximo passo foi escrever uma extensão para realizar essa função. Uma extensão em Python é um módulo implementado na mesma linguagem do interpretador. No caso, como foi utilizada a implementação em C de Python, a extensão poderia ser escrita tanto em C quanto em C++. Optou-se por C, já que algumas funções da GDK seriam necessárias. Já que a GDK foi utilizada, também foram usadas várias funções da Glib, de modo que o código fosse o mais portável possível. A extensão apresentou desempenho aceitável, sendo cerca de 3 vezes mais rápida que a implementação em Python utilizando NumPy.

4. Considerações e Conclusões

Existem outras opções, além das apresentadas. Uma das mais interessantes, considerando futuras possibilidades, chama-se Psyco. É um compilador JIT (Just In Time) para Python, que é muito útil para aplicações que utilizem principalmente a CPU (CPU-Bound), mas pouco útil para aplicações com muita Entrada/Saída de Dados (I/O-Bound). Atualmente Psyco não tem desenvolvimento ativo, pois os esforços concentram-se no PyPy, ou Python in Python, um compilador para Python escrito em Python, que visa prover no futuro vários backends diferentes para a compilação de código em Python (indo desde a LLVM, até Assembly, passando pela JVM e pela CLR). PyPy encontra-se atualmente em estágio de desenvolvimento e pesquisa, e uma versão pronta para ambientes de produção está prevista para o fim de 2007.

O principal objetivo desse artigo foi mostrar que, utilizando as ferramentas corretas, Python é uma boa opção para a implementação de aplicações científicas, principalmente por facilitar a implementação de novas funcionalidades após o sistema estar consolidado e por prover um rápido ambiente para a prototipação de soluções que podem ser adotadas sem a necessidade de uma reescrita de código para outras linguagens de programação.

Referências

NumPy (2007). Numpy. Disponível em `http://numpy.scipy.org`. Acesso em: 19/02/2007.

Ramachandran, P. (2007). Performance python. Disponível em `http://www.scipy.org/PerformancePython`. Acesso em: 19/02/2007.