UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUÍS FELIPE GARLET MILLANI

Análise de Correlação Entre Métricas de Qualidade de Software e Métricas Físicas

Trabalho de Graduação

Prof. Dr. Antônio Carlos Beck Filho
Orientador

Prof. Dr. Luigi Carro
Co-orientador

Porto Alegre, janeiro de 2013

AGRADECIMENTOS

Agradeço aos meus pais por todo incentivo que sempre me deram.

Agradeço ao meu irmão por sempre me apoiar.

Agradeço ao Ulisses Brisolara Corrêa pelas suas contribuições, sem as quais este trabalho não teria sido realizado.

Agradeço ao meu orientador, prof. Dr. Antônio Carlos Beck Filho, por toda ajuda e por sempre ter paciência.

Agradeço ao meu co-orientador, prof. Dr. Luigi Carro, que me acompanhou no início deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CBO *Coupling Between Objects*

DF *Direction Flag*

DIT *Depth of Inheritance Tree*

DI *Destination Index register*

ESP *Extended Stack Pointer*

ES *Extra Segment*

FDR *False Discovery Rate*

IPC *Instructions Per Cycle*

ISA *Instruction Set Architecture*

LCOM *Lack of Cohesion in Methods*

MSE *Mean Squared Error*

SHA *Secure Hash Algorithm*

SPEC *Standard Performance Evaluation Corporation*

SSE *Sum of Squared Errors*

SSH *Secure Shell*

SSL *Secure Sockets Layer*

XML *Extensible Markup Language*

XSLT *Extensible Stylesheet Language Transformations*

LISTA DE FIGURAS

RESUMO

Devido ao aumento de complexidade do software embarcado, seus custos de desenvolvimento e manutenção estão em ascensão. Além da crescente complexidade, têm que ser observados os limites de desempenho, consumo de memória e consumo de potência, limites estes que, diferentemente do software tradicional, normalmente não podem ser ignorados durante o seu desenvolvimento.

Com o uso de orientação a objetos pode-se amenizar essas dificuldades, e, também, se obter software de melhor qualidade. No entanto, essa solução muitas vezes não é adotada para o software embarcado, pois supõe-se que o impacto da orientação a objetos no desempenho, consumo de memória e consumo de potência não permita atender aos requisitos de projeto. Contudo, não se sabe com certeza como a execução de um software é influenciada pela maneira como é escrito.

Neste contexto, este trabalho de conclusão tem como objetivo propor uma metodologia para verificar se e como a forma de escrita do software, com respeito a orientação a objetos, impacta na sua execução. Para avaliar a forma de escrita do software de forma objetiva, utilizaram-se métricas de qualidade de software, que permitem discretizar diversas características do código fonte. Com isso é possível analisar a existência de correlação entre o código escrito e seu desempenho em diferentes configurações de uma mesma arquitetura. Contudo, não foi possível encontrar correlação entre as métricas de qualidade de software e as métricas físicas.

Palavras-chave: Sistemas embarcados, engenharia de software, métricas de qualidade de software, métricas físicas, simulação.

Correlation Analysis Between Software Quality Metrics and Physical Metrics

ABSTRACT

Because of the growing complexity of embedded software, its development and maintenance costs are in ascension. Besides the increasing complexity, there are also physical limits that must be obeyed, like performance, memory use and power consumption. While in traditional software these limits can often be ignored, the same is not true for embedded software.

One possibility to ease the difficulties that arise from high complexity is to use object orientation, which results in software of better quality. However, this solution is often not adopted for embedded software as the impact of object orientation in performance, memory use and power consumption is assumed to be too high to satisfy the project's requirements. Although that is assumed, it is not known how the execution of a software is influenced by how it is written.

In this context, this work's objective is to propose a methodology to verify if and how the way software is written, with respect to object orientation, impacts in its execution. To measure the many characteristics of the source code in an objective way, software quality metrics are used. This allows to analyse the existence of a correlation between the way software is written and how it performs in different configurations of a single architecture.

Keywords: embedded systems, software engineering, software quality metrics, physical metrics, simulation.

1 INTRODUÇÃO

O software embarcado sempre desempenhou tarefas bem mais simples que o software tradicional. Contudo, como consequência do crescente uso de dispositivos embarcados para desempenhar tarefas historicamente realizadas por computadores pessoais, a complexidade do software embarcado tem aumentado consideravelmente. Além de requisitos funcionais cada vez mais abrangentes, há também a necessidade de atender a limites de potência, consumo energético, processamento e ocupação de memória - limites estes que frequentemente podem ser ignorados em computadores pessoais.

Softwares complexos possuem elevado tempo e custo de desenvolvimento, além de serem de difícil manutenção. Uma forma de amenizar essas consequências negativas é com maior uso de orientação a objetos (ABREU; MELO, 1996). Todavia, mesmo apresentando diversas vantagens, essa opção tende a ser evitada pois pressupõe-se que resulta em baixo desempenho. Orientação a Objetos é associada a baixo desempenho, quando comparada com abordagens menos abstratas. No entanto, não existem estudos conclusivos que mostrem se esse paradigma é, de fato, mais custoso e, caso positivo, o quão significativos são esses custos nas diferentes arquiteturas empregadas nos sistemas embarcados.

Um melhor entendimento de como as métricas de qualidade de software influenciam as métricas físicas (características como consumo energético, potência, ocupação de memória, instruções executadas e instruções por ciclo) do mesmo facilitaria a escrita de software capaz de atender requisitos físicos com um menor comprometimento da qualidade do código. Nesse contexto, propõe-se aplicar métricas de software de forma a caracterizar o emprego da Orientação a Objetos em diferentes implementações de softwares representativos de sistemas embarcados. Dentre as características de software analisadas através das métricas podemos citar: acoplamento, coesão, herança, etc. O uso de Orientação a Objetos tende a levar a soluções mais simples que alternativas de mais baixo nível, e, portanto, influencia, também, outras métricas. Assim, algumas dessas métricas também são analisadas neste trabalho, dentre elas: complexidade ciclomática, profundidade de blocos aninhados, etc.

Este trabalho tem como objetivo propor uma metodologia para verificar como diversas características do software, mensuradas a partir de métricas de software, impactam no seu desempenho, estimado através da execução de simulações. As métricas físicas e de qualidade são analisadas com métodos estatísticos implementados no Matlab. Saber como a forma de escrita do código fonte influencia na execução do aplicativo possibilitaria ao desenvolvedor tomar decisões de projeto mais bem informadas, resultando em melhor qualidade de software e, ao mesmo tempo, atendendo aos limites físicos.

2 METODOLOGIA

A metodologia proposta consiste em executar um conjunto de *benchmark* em um simulador, de forma a obter métricas físicas, como Instruções por Ciclo (IPC) e em um *profiler*, para obter um perfil de execução, que contém uma lista dos métodos executados. Utiliza-se um extrator de métricas no código fonte e, com isso, obtém-se as métricas de qualidade de software. As métricas de qualidade de software são, então, filtradas de forma a remover os métodos não utilizados, com auxílio do perfil de execução. As métricas físicas e as métricas de qualidade são, então, analisadas de forma a verificar se, estatisticamente, estão correlacionadas. Esse processo está ilustrado na Figura 2.1.

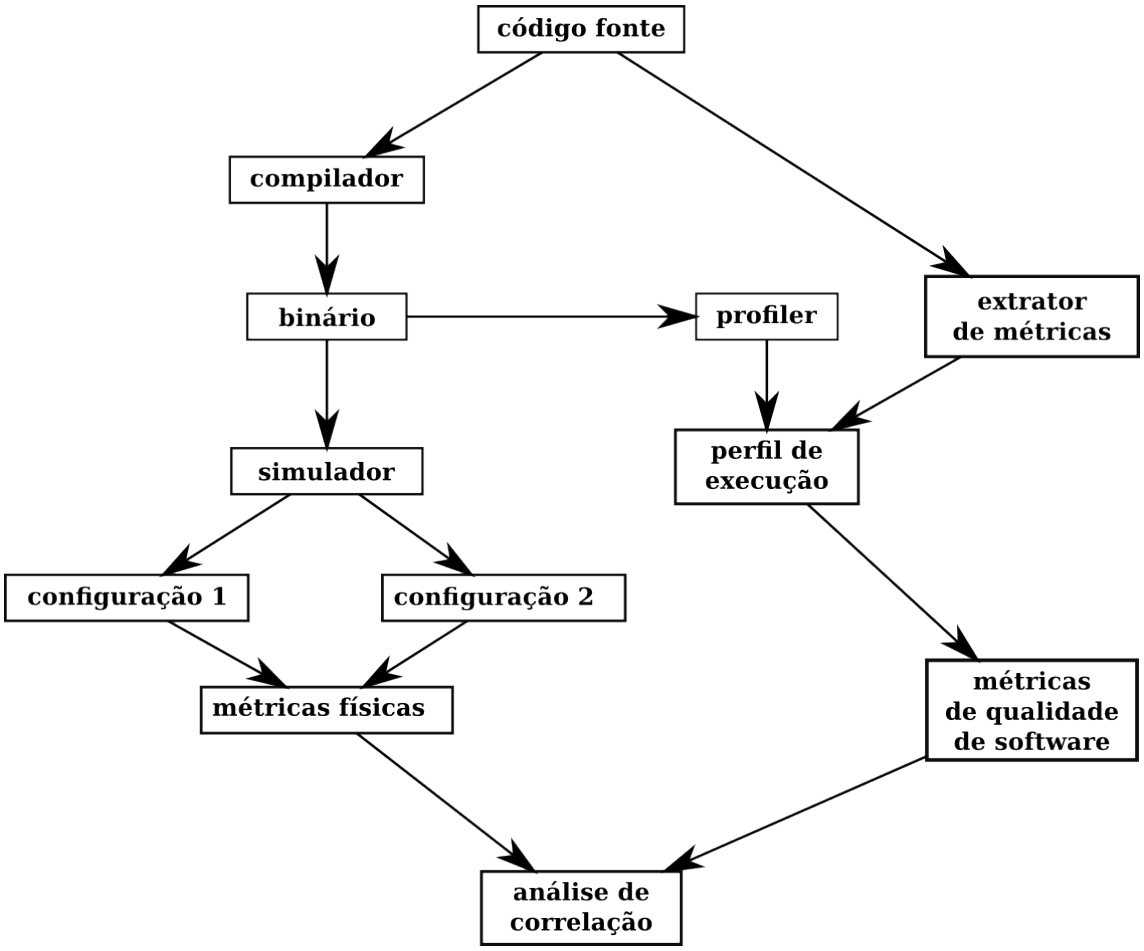


Figura 2.1: Metodologia.

2.1 Seleção dos *Benchmarks*

As aplicações selecionadas devem ser representativas do conjunto sobre o qual deseja-se fazer inferências estatísticas. Contudo, enquanto a escolha do conjunto de aplicações a serem analisadas é de alta importância, isso não é suficiente para ter-se *benchmarks* representativos - as entradas utilizadas nas execuções das aplicações podem influenciar significativamente o perfil das execuções. Assim, é necessário que tenha-se o cuidado de selecionar, também, entradas representativas.

No caso dos *benchmarks* selecionados neste trabalho, o intuito é que sejam representativos do conjunto de aplicativos utilizados em sistemas embarcados. Assim, utilizaram-se diferentes aplicativos de criptografia, codificação e decodificação de vídeo e imagens, navegação web e análise e validação XML. Essas categorias foram selecionadas baseando-se nos benchmarks que compõem o SPEC JVM 2008, que é descrito na Seção 3.2.

2.2 Métricas Físicas

As métricas físicas de um sistema de software (conjunto formado por hardware e software) refletem os aspectos externos do software - como consumo energético, potência, desempenho e ocupação de memória. Como sistemas embarcados possuem limites rígidos de consumo, potência, desempenho e memória, as métricas físicas são bastante úteis para verificar se o sistema de software atende ou não aos requisitos de projeto.

Neste trabalho limitou-se a análise a uma única métrica física, chamada Δ IPC (Δ Instruções Por Ciclo), devido a limitações das ferramentas. Como as duas configurações utilizadas diferem apenas no tamanho do pipeline, valores altos de Δ IPC significam melhor uso do pipeline. O Δ IPC é definido, para cada *benchmark* K, como a variação de IPC entre duas diferentes configurações de uma mesma ISA (*Instruction Set Architecture*):

$$IPC_{KA} : \text{IPC do } \textit{benchmark } K \text{ executando na configuração } A$$

$$IPC_{KB} : \text{IPC do } \textit{benchmark } K \text{ executando na configuração } B$$

$$\Delta IPC_K = IPC_{KA} - IPC_{KB}$$

Métricas físicas podem ser obtidas de diferentes formas, como execução nativa ou execução simulada. Essas duas possibilidades são descritas a seguir, bem como algumas de suas vantagens e desvantagens.

2.2.1 Execução nativa

Uma possibilidade é executar os *benchmarks* nativamente e medir, através de ferramentas comumente disponibilizadas pelo sistema operacional, dados como tempo de execução e ocupação de memória. No GNU/Linux tem-se o utilitário *time*[1], que mede o tempo total de execução, o tempo de utilização da CPU pelo usuário e tempo de utilização da CPU pelo sistema (por exemplo para chamadas de sistema como leitura de arquivos). Também tem-se ferramentas como o *perf*[2], que, além de contabilizar o tempo de execução, disponibiliza informações como número de instruções e desvios. No Windows o *Timeit*, incluído no Windows Server 2003 Resource Kit[3] possui características similares ao *time*. Algumas vantagens da execução nativa:

[1] http://linux.die.net/man/2/time

[2] http://linux.die.net/man/1/perf-stat

[3] https://www.microsoft.com/en-us/download/details.aspx?id=17657

Facilidade de uso Como as únicas ferramentas utilizadas já fazem parte do sistema operacional, é mais fácil e simples de ser feita;

Tempo de execução A coleta de dados de um programa executado nativamente possui baixo *overhead*.

Dentre as desvantagens da execução nativa, temos:

Dados limitados Os dados disponibilizados normalmente são bastante limitados. Com o *time*, por exemplo, tem-se apenas os tempos de execução. Dados como *cache misses* ou número de instruções executadas não podem ser facilmente obtidos. Os utilitários que obtêm esse tipo de informações, como o *perf*, fazem-no por meio de *sampling*, o que resulta em resultados aproximados. Dados de consumo energético podem ser medidos com equipamento específico;

Pouco controle Tem-se pouco controle sobre características físicas da máquina. Qualquer mudança - aumento de cache L1, por exemplo, requer modificar o hardware da máquina, o que pode não ser viável;

Dependência do hardware Os resultados obtidos são específicos para o hardware que os executou, ou seja, para obter resultados de diferentes configurações de hardware é necessário possuir um computador com a configuração que deseja-se mensurar.

2.2.2 Simulação

Uma alternativa à execução nativa é a execução simulada. Nesse caso a execução do *benchmark* é simulada, ou seja, não é feita diretamente pelo hardware, mas sim por um aplicativo que copia as várias características do hardware simulado a nível estrutural (simulando, por exemplo, um pipeline mas desconsiderando as características físicas do circuito). Como a execução é feita em software, pode-se simular diferentes arquiteturas, inclusive arquiteturas que existem apenas teoricamente. Algumas vantagens da execução simulada:

Diversidade de dados Como um dos objetivos dos simuladores é a obtenção de dados da execução, pode-se obter um grande número de dados. Por exemplo, número de ciclos, número de chamadas de sistema, percentual de erros no preditor de desvio e número de cache misses. Contudo, alguns dos dados podem ser estimativas - como consumo energético, caso estimado pelo simulador;

Controle total Dependendo do simulador, pode-se ter controle total sobre os detalhes da simulação. É possível simular diferentes arquiteturas e diferentes configurações;

Independência do hardware Como os resultados não dependem do hardware em que executam, pode-se executar os *benchmarks* de uma única configuração em computadores diferentes.

Algumas desvantagens da execução simulada:

Dificuldade de uso Longo tempo de execução dificulta usar aplicações interativas. Além disso, o simulador não necessariamente implementa todas instruções da ISA simulada, o que significa que nem todo binário que pode ser executado nativamente pode ser executado no simulador, podendo ser necessário recompilá-lo de forma a remover as instruções não suportadas;

Tempo de execução Dependendo das características do simulador utilizado, o *overhead* da simulação pode ser bastante alto. Por exemplo, um *benchmark* que demora poucos segundos para executar nativamente pode demorar minutos, ou mesmo horas, para ser simulado.

Devido à simulação possibilitar a obtenção de métricas físicas em diferentes configurações da arquitetura e por disponibilizar dados mais precisos do que os obtidos com execução nativa, optou-se por utilizar um simulador. O simulador escolhido foi o Multi2Sim (UBAL et al., 2012), que simula a arquitetura x86.

2.3 Métricas de Qualidade de Software

Métricas de qualidade de software são uma maneira de discretizar características do código fonte, sem necessitar que uma pessoa leia e interprete-o. Mesmo existindo pouca dúvida de que alguma forma de medir a qualidade do software é necessária, não há consenso sobre o quão precisas são as métricas (LINCKE; LUNDBERG; LÖWE, 2008).

Tipicamente, utilizam-se métricas de qualidade de software para rapidamente avaliar quais seções do código precisam ser refatoradas ou quais têm mais probabilidade de possuírem *bugs* (SIMON; STEINBRUCKNER; LEWERENTZ, 2001). Enquanto métricas de qualidade de software são utilizadas por várias organizações, seu uso ainda não é muito difundido no desenvolvimento de software em geral, sendo restrito a softwares altamente complexos e/ou com requerimentos bastante estritos.

2.3.1 Obtenção das Métricas de Qualidade de Software

As métricas podem ser obtidas através do uso de uma ou mais ferramentas de extração de métricas. Existem várias ferramentas para tal, contudo não existe uma ferramenta que seja claramente melhor que todas as outras - todas possuem pontos fortes em uma área e fracos em outra. Suporte a diferentes linguagens de programação é frequentemente limitado, bem como quais métricas são calculadas. Assim, a ferramenta a ser utilizada depende da linguagem na qual os *benchmarks* foram escritos e, também quais métricas deseja-se obter. Apesar disso, escolhida a ferramenta, a obtenção de métricas da mesma é relativamente fácil. Neste trabalho utilizou-se a ferramenta Understand, que é descrita na Seção 4.1.5.

2.4 Análise de Correlação

A partir dos dados das métricas físicas e métricas de qualidade de software, pode-se verificar se há correlação entre as duas. Ou seja, verifica-se, estatisticamente, quão bem é possível explicar as métricas físicas com base nas métricas de qualidade. A análise de correlação é vista com mais detalhes na seção 6

3 BENCHMARKS

Os *benchmarks* foram selecionados a partir de aplicações reais e sintéticas de forma a abranger diversas funções importantes de dispositivos móveis - criptografia, compressão, codificação e decodificação de vídeo e imagens, navegação web e *XML parsing*, abrangindo diversas das categorias que compõem o SPEC JVM 2008, que é descrito na Seção 3.2. Devido à extração de métricas requerer acesso ao código fonte, somente programas de código aberto foram utilizados. Como disponibilidade de acesso ao código fonte é mais comum em programas para GNU/Linux, optou-se por limitar os programas analisados a esse sistema operacional. Além disso, para facilitar a simulação, selecionou-se apenas aplicativos que não requerem interação com o usuário.

3.1 Aplicativos

Diversos dos aplicativos selecionados para *benchmark* foram escritos em C devido a dificuldade de encontrar aplicativos em C++ que atendam às limitações citadas anteriormente. Esses aplicativos são apresentados na Tabela 3.1. Cada um deles é discutido a seguir.

Categoria	Algoritmo
Criptografia	AES, Blowfish, DES, RSA, SHA, certificado digital
Compressão	DEFLATE
Descompressão	DEFLATE
Codificação de vídeo	H.263, H.264, MPEG-1, MPEG-2
Decodificação de vídeo	H.263, H.264, MPEG-1, MPEG-2
Codificação de imagens	JPEG
Decodificação de imagens	JPEG
Navegação web	HTML
XML	análise, validação

Tabela 3.1: Benchmarks utilizados.

Crypto++[1] Biblioteca escrita em C++ que implementa diversos algoritmos de criptografia. Para utilizar a biblioteca neste trabalho foram desenvolvidos quatro programas, que usam diferentes funcionalidades dessa:

AES O Advanced Encryption Standard (Advanced Encryption Standard (AES), 2001), ou Rijndael, é uma cifra de bloco criada por Joan Daemen e Vincent

[1] http://www.cryptopp.com

Rijmen em 1998 e adotada pelo governo dos Estados Unidos em 2001 como sucessora do DES. Uma cifra de bloco é um algoritmo que criptografa uma sequência de blocos, ou grupos, de bits. O algoritmo utiliza chave simétrica, ou seja, uma única chave para criptografar e decriptografar. Pode ser usado para criptografar qualquer sequência de blocos de 128 bits. É um dos algoritmos de chave simétrica mais utilizados, possuindo inclusive instruções específicas, implementadas em alguns processadores da AMD, Intel, SPARC e ARM. Entre outros usos, a cifra é utilizada para criptografia de disco e comunicação;

DES O Data Encryption Standard é uma cifra de bloco de chave simétrica publicada em 1977 pela IBM. Como a chave tem apenas 56 bits, o algoritmo é hoje considerado inseguro. Em 1998 surgiu uma modificação do algoritmo, chamada Triple DES, que executa o DES três vezes em cada bloco. O Triple DES ainda é bastante utilizado por sistemas bancários e é considerado seguro pelo *National Institute of Standards and Technology* dos Estados Unidos (BARKER, 2004);

RSA O RSA (a sigla vem do nome dos autores: Ron Rivest, Adi Shamir e Leonard Adleman) é um algoritmo de chave pública publicado em 1977 (BELLARE; ROGAWAY, 1996). Um algoritmo de chave pública utiliza duas chaves, uma que deve ser mantida em segredo - a chave privada - e outra que pode ser de conhecimento público. A geração das chaves requer a fatoração de números inteiros grandes, o que é um processo computacionalmente custoso. As mensagens criptografadas com a chave pública somente podem ser decriptadas pela chave privada, e as criptografadas com a chave privada somente podem ser decriptadas pela chave pública. O algoritmo possui diversos usos, como, por exemplo, fazer login em SSH (*Secure Shell*) por meio de chaves ao invés de senha e em SSL (*Secure Sockets Layer*), que é um protocolo utilizado para criptografar as mais variadas aplicações, como navegação web, e-mail e voz por IP;

Sign/Verify Este programa visa implementar assinatura e verificação de assinaturas digitais. Uma assinatura digital permite garantir autenticidade (o receptor tem como saber que foi, de fato, o emissor que assinou a mensagem), integridade (o receptor tem como confirmar que a mensagem não foi alterada) e irretratabilidade (após assinada, o emissor não pode remover sua assinatura). Assinaturas digitais tem vários usos, o mais comum sendo em certificados digitais, que são usados para que o usuário possa verificar, por exemplo, que quando ele acessa o site do seu banco é realmente o banco e não alguém mal-intencionado. O programa em questão faz duas assinaturas e verificações: RSA/SHA-1 e RSA/SHA-256. A diferença entre as duas é apenas o número de bits da função de *hash* SHA, o SHA-1 com 160 bits e o SHA-256 com 256 bits. Segundo (BELLARE; ROGAWAY, 1996) o processo de assinar e verificar uma mensagem usando o RSA é feito da seguinte maneira: para assinar uma mensagem calcula-se o seu *hash* (com SHA-1 ou SHA-256) e, então, aplica-se o RSA, com a chave privada, sobre o *hash* calculado. Para verificar uma assinatura o processo é similar: calcula-se o hash da mensagem e utiliza-se a chave pública na assinatura disponibilizada. Caso o resultado seja igual ao *hash* calculado a assinatura é valida;

gzip[2] O gzip é um compactador e descompactador de arquivos criado em 1992 como um substituto do utilitário compress de Unix para ser usado no GNU Project. O aplicativo é escrito em C e utiliza o algoritmo DEFLATE. Apesar de bastante antigo, é um dos compressores mais utilizados para distribuição de código fonte por requerer pouco processamento e memória;

links Navegador web em modo texto (existe um modo gráfico mas esse não foi utilizado) escrito em C. Assim como a maior parte dos navegadores que não suportam interface gráfica, não há suporte a *plugins* como Flash, nem JavaScript. Ainda assim, o suporte a HTML 4.0 é relativamente bom, tendo melhor suporte a tabelas que outros navegadores similares, como o Lynx. Optou-se por um navegador em modo texto devido a navegadores com interface gráfica, como Chrome, Internet Explorer, Firefox, Opera e Safari requererem interação com o usuário, o que é difícil de ser feito ao executar numa simulação, já os navegadores com interface por texto podem ser facilmente executados sem interação com o usuário;

MediaBench II[3] Suíte de *benchmarks* escritos em C voltados à codificação e decodificação de vídeos e imagens. Foram utilizados todos os aplicativos da suíte:

- **cjpeg** O cjpeg é um codificador JPEG. O formato JPEG é um dos métodos de compressão com perda de qualidade mais usados para fotos e imagens em geral. Por ser um formato com perdas, ao comprimir-se uma imagem alguns dados são perdidos, ou seja, a imagem descomprimida não é idêntica à original. Mas, por outro lado, o tamanho em bytes da imagem é bastante reduzido - o que facilita o armazenamento e, principalmente, o compartilhamento de imagens. Em dispositivos móveis com capacidades de armazenamento e de transferência de dados bastante limitadas, o uso de compressão JPEG permite armazenar e transmitir um número muito maior de imagens do que seria possível caso as imagens não fossem compactadas;
- **djpeg** O djpeg é um decodificador JPEG. Além de necessário em dispositivos com câmera para visualizar as imagens armazenadas, descomprimir JPEGs é importante também para navegação web devido ao grande número de imagens JPEG utilizados na web;
- **h263dec** O h263dec é um decodificador de vídeos H.263. O formato H.263 foi criado com o intuito de ser utilizado para video-conferências. Com a disseminação de acesso à Internet por banda larga, o formato passou a ser utilizado em vídeos em Flash em sites como YouTube.com. Mesmo com o surgimento de formatos com melhor qualidade de compressão, como o H.264, o H.263 continua em uso por ser computacionalmente menos custoso;
- **h263enc** O h263enc é um codificador de vídeos H.263. É usado em dispositivos que gravam vídeo e possuem pouco espaço de armazenamento e processamento limitado;
- **h264dec** É um decodificador de vídeos H.264. O formato H.264 é um dos mais utilizados hoje para vídeos em alta definição. Grande parte dos websites de transmissão de vídeos, como YouTube, Vimeo e Hulu, utilizam o formato;

[2] http://www.gzip.org
[3] http://euler.slu.edu/ fritts/mediabench

h264enc Codificador de vídeos H.264. Possui custo computacional maior que o H.263 mas possui maior taxa de compactação;

jpg2000dec Decodificador JPEG similar ao djpeg;

jpg2000enc Codificador JPEG similar ao cjpeg;

mpeg2dec Decodificador do padrão MPEG-2, que é uma especificação de vários formatos de compressão de vídeo e áudio. O MPEG-2 inclui os formatos de compressão de vídeo do MPEG-1 e, também H.262, bem como novos padrões de compactação de áudio. É o padrão utilizado em DVD-vídeo, e um dos padrões utilizados para televisão digital e por satélite;

mpeg2enc Codificador MPEG-2;

mpeg4dec Decodificador do padrão de MPEG-4, que inclui diversos formatos de compressão de vídeo e áudio. Foi criado com base no MPEG-2 e, além dos padrões especificados por esse inclui o formato de compressão de vídeo H.264.

Além destes, foi utilizada uma versão portada de C para C++ do mpeg2dec;

mpeg1 Implementação em C de decodificação de áudio MPEG-1. O MPEG-1 é um dos padrões de decodificação mais utilizados devido ao suporte a áudio MP3, que existe em um grande número de dispositivos;

MiBench Suite de *benchmarks* escritos em C e voltados a sistemas embarcados (GUTHAUS et al., 2001). Foram utilizados os seguintes *benchmarks*:

blowfish Blowfish é um algoritmo de cifra de bloco com chave simétrica, assim como o AES. Foi introduzido em 1993 como uma alternativa ao DES. O algoritmo é um dos mais usados, contudo perde em popularidade para o AES;

rijndael Implementa o algoritmo de criptografia AES;

sha Implementa o algoritmo de *hash* Secure Hash Algorithm (SHA). A versão implementada é o SHA-1, que, segundo (Secure Hash Standard, 2012), possui 160 bits, ou seja, é uma função que mapeia a entrada (um ou mais blocos de 512 bits) para inteiros de 160 bits. Algoritmos de *hashing* possuem vários usos, como, por exemplo, verificar integridade de arquivos e verificação de assinaturas;

Sablotron[4] Ferramenta para processamento de XML. O Sablotron foi escrito em C++ mas pode ser usado por aplicativos feitos em outras linguagens, como C, Perl, Python, PHP e ObjectPascal. A ferramenta suporta Extensible Stylesheet Language Transformations (XSLT), que é uma linguagem que permite transformar documentos XML para diferentes formatos. O XSLT permite, por exemplo, a partir de dados em XML gerar uma página HTML ou um documento PDF. Além disso, pode-se utilizar de XSLT para converter os dados XML de uma aplicação para outro XML que possa ser utilizado em uma aplicação diferente;

Xerces[5] O Xerces é uma biblioteca de XML desenvolvida pela Apache Software Foundation. O Xerces implementa diferentes APIs para análise XML, como Document Object Model (DOM) e Simple API for XML (SAX). Por ser uma biblioteca, foi necessário escrever um pequeno programa para utilizar as funções implementadas.

[4] http://www.gingerall.com/charlie/ga/xml/p_sab.xml
[5] https://xerces.apache.org/xerces-c

3.2 Entradas

Como a entrada utilizada pode influenciar significativamente na execução dos *benchmarks*, sempre que possível foram utilizadas as entradas disponibilizadas pela Standard Performance Evaluation Corporation (SPEC). A SPEC é uma organização que desenvolve conjuntos de *benchmarks* com diferentes funções, como verificar performance de operações com números inteiros ou ponto flutuante, virtualização, etc. Mais especificamente, foram utilizadas as entradas do conjunto de *benchmarks* conhecido como SPEC JVM 2008[6], que contém *benchmarks* computacionalmente intensivos, com pouco uso de I/O e sem uso de comunicação por rede entre diferentes computadores. O SPEC JVM 2008 inclui programas escritos em Java, contudo esses não foram utilizados.

Quando mais de uma entrada foi utilizada, ou seja, nos casos em que o aplicativo foi executado mais de uma vez, utilizou-se a média dos resultados (isto é, para o caso de IPC, somou-se o valor de IPC para cada entrada e dividiu-se pelo número de entradas). Nos casos em que não havia entrada apropriada no SPEC JVM 2008, como na codificação e decodificação de imagens e vídeos, utilizaram-se as entradas para teste incluídas com o código fonte.

3.3 Compilação

Para os programas C foi utilizada a versão 4 do compilador gcc, enquanto que, para os em C++, foi utilizada a versão 4 do compilador g++. Como grande parte dos softwares utilizados são de GNU/Linux, somente foram criados binários para esse sistema operacional. Os binários foram compilados estaticamente e para a arquitetura i386.

De forma a reduzir a influência do compilador, todos os programas foram compilados com otimizações desabilitadas, isto é, o compilador gerou o código da forma mais simples, sem tentar torná-lo mais eficiente. Isso pode afetar os resultados, uma vez que normalmente otimizações são utilizadas. Contudo, habilitar otimizações também poderia produzir resultados tendenciosos, pois otimizações são bastante sensíveis, a ponto que pequenas mudanças na ordem da aplicação de otimizações pode afetar drasticamente o executável gerado (KULKARNI et al., 2003). Ou seja, o uso de otimizações dificultaria diferenciar quais resultados são consequência das otimizações e quais são consequência de como o código foi escrito. Além disso, o uso de otimizações pode resultar em usar uma maior gama das instruções disponíveis - e como o simulador somente implementa algumas das instruções, a simulação seria dificultada.

[6] http://www.spec.org/jvm2008

4 MÉTRICAS DE QUALIDADE DE SOFTWARE

4.1 Extrator de Métricas

Um extrator de métricas, ou coletor de métricas, é um software que analisa o código fonte de uma aplicação e, a partir dos dados coletados, calcula diversas métricas de qualidade de software. Os seguintes extratores de métricas foram testados: Analizo e Kalibro Metrics, CCCC, Dependometer, Sonar e Understand. Por fim, um extrator foi escolhido.

4.1.1 Analizo e Kalibro Metrics

O Analizo[1] é uma ferramenta de extração de métricas inicialmente baseada no egypt[2], que é uma ferramenta para geração de grafos de chamadas[3]. Apesar do código das primeiras versões ser bastante similar ao do egypt, a versão atual do Analizo é bastante diferente - o Analizo faz também o cálculo de várias métricas. É uma ferramenta feita para ser utilizada por ferramentas que permitam visualização das métricas, portanto, o Analizo disponibiliza os resultados no formato YAML, cujo objetivo é facilitar o acesso por outras ferramentas, mas que permite que o usuário leia os resultados. Por ser escrita em Perl, é uma ferramenta bastante portável, possuindo suporte para GNU/Linux, Mac OSX e Windows. As linguagens suportadas são: C, C++ e Java.

A análise do código fonte é feita através do Doxyparse[4], que é uma ferramenta desenvolvida com base no Doxygen (o Doxygen, por sua vez, é um gerador de documentação a partir do código fonte). O Doxyparse tem a função de analisar o código fonte e extrair informações como declarações e usos de funções.

Um dos problemas da ferramenta é o pequeno número de métricas suportadas. No total, 24 métricas são calculadas. Dessas, algumas são métricas de pouco interesse tendo em vista a predição de características físicas e não do código em si (comentários, por exemplo, não influem no código executado). Algumas das métricas são interessantes, como Falta de Coesão entre Métodos e Profundidade da Árvore de Herança.

Junto do Analizo foi utilizado o Kalibro Metrics[5] (Figura 4.1), que é uma ferramenta desenvolvida pelo Centro de Competência em Software Livre da USP que para ser utilizada em conjunto com o Analizo ou outro extrator de métricas. A função da ferramenta é facilitar a interpretação das métricas coletadas, inclusive permitindo a criação de limites que, quando atingidos, geram um aviso ao usuário sobre quais partes do código devem

[1] http://analizo.org

[2] http://www.gson.org/egypt/

[3] Um grafo de chamada é um grafo direcional que mostra as relações entre os métodos de um programa da seguinte forma: os nodos são métodos, e um vértice (A, B) significa que o método A chama o método B.

[4] http://ccsl.ime.usp.br/files/Analizo-InstallationGuide.pdf

[5] http://ccsl.ime.usp.br/kalibro

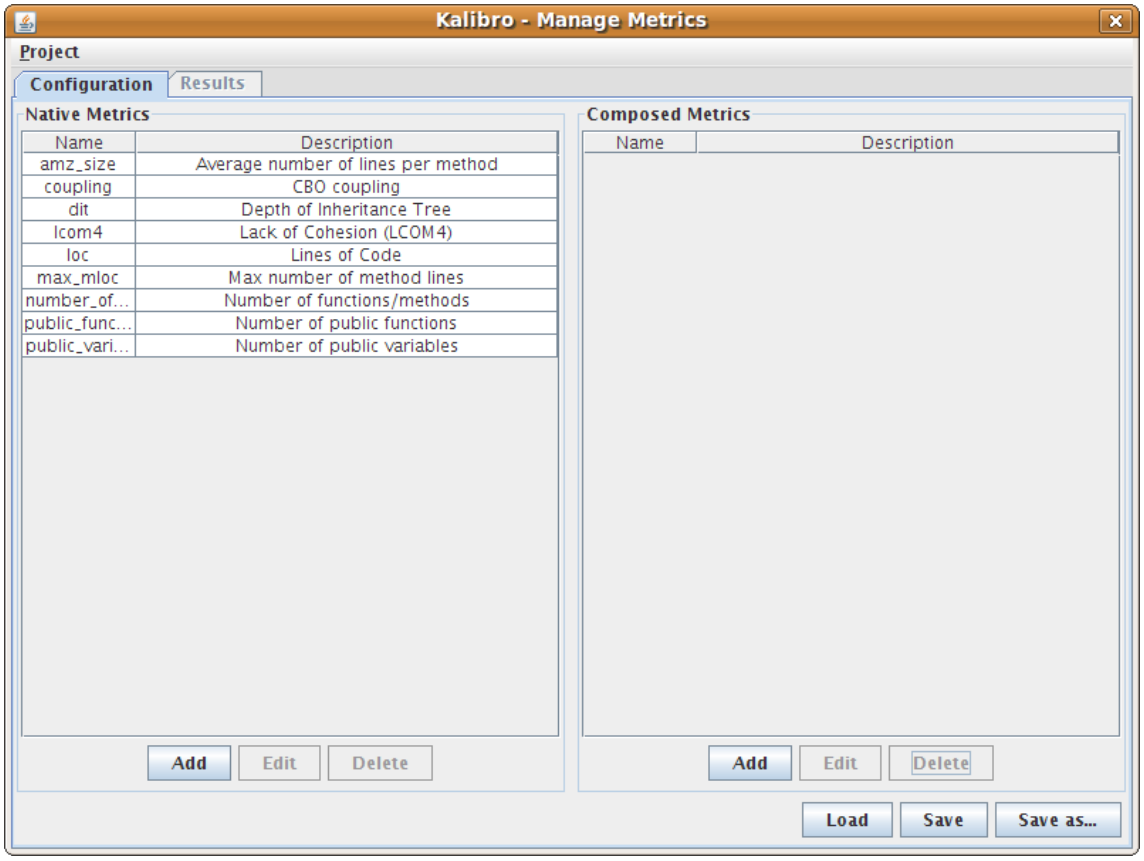


Figura 4.1: Kalibro Metrics.

ser refatoradas.

Uma característica interessante da ferramenta é facilitar o cálculo de novas métricas pelo usuário da ferramenta com código JavaScript. O Kalibro Metrics permite a exportação dos dados para o formato CSV (comma-separated values), o que facilita acesso a partir de outras ferramentas pois suporte a CSV é bastante comum.

4.1.2 Dependometer

O Dependometer[6] (Figura 4.2), além de ser um extrator de métricas também analisa as dependências entre as diversas camadas, subsistemas e classes do código fonte analisado. As linguagens suportadas são C++, C# e Java. Como a ferramenta é feita em Java, suporta os sistemas operacionais GNU/Linux, Mac OSX e Windows.

Uma diferença do Dependometer para as outras ferramentas de métricas analisadas é requerer a criação de um arquivo de configuração XML para cada projeto analisado. Isso torna o uso da ferramenta mais difícil que as outras, que não requerem leitura da documentação para tarefas simples.

Como o objetivo da ferramenta não é exclusivamente a extração de métricas, as métricas disponibilizadas são poucas - apenas 17, desconsiderando as métricas consideradas triviais como número de linhas de código ou linhas em branco. Mesmo com algumas dessas sendo métricas interessantes, como acoplamento eferente e acoplamento entre objetos, o número de métricas não-triviais disponibilizadas é pequeno.

[6]http://source.valtech.com/display/dpm/Dependometer

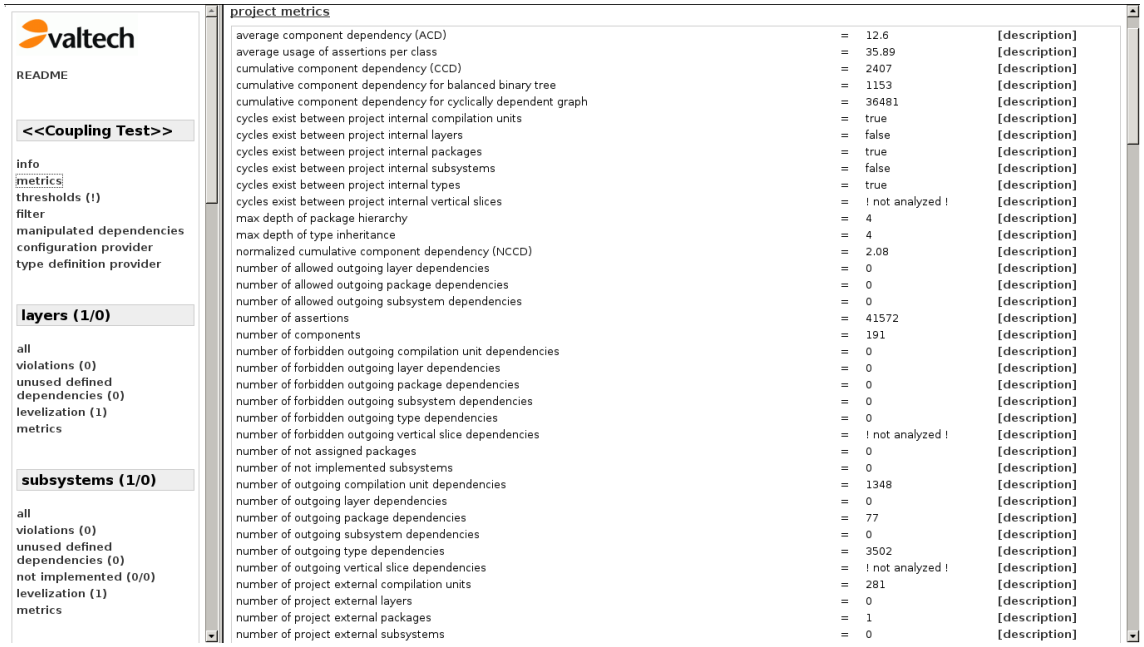


Figura 4.2: Dependometer, com o código do Crypto++.

4.1.3 Sonar

O Sonar[7] (Figura 4.3) é uma plataforma de qualidade de software que utiliza diversas ferramentas para obter métricas de software. A ferramenta é escrita em Java e possui integração com ferramentas comumente utilizadas na plataforma Java, como Eclipse e Maven. O sistema de *plugins* da ferramenta facilita a expansão das suas funcionalidades e permite suporte a múltiplas linguagens, dentre essas ABAP, C, C++, C#, Cobol, Delphi, Pascal, Drools, ActionScript, Groovy, Java, JavaScript, Natural, Pacbase, PHP, PL/I, PL/SQL, Python, VB.NET, VB.NET, Visual Basic e XML.

Como várias das métricas são calculadas a partir de *plugins*, e a maior parte desses é específica para uma única linguagem, as métricas disponíveis dependem da linguagem utilizada. Não incluindo as métricas de teste, Java, por exemplo, possui 40 métricas, enquanto C++ possui apenas 19 e C somente 17. Além do número relativamente pequeno de métricas, poucas das métricas foram consideradas interessantes, sendo que a maioria considera apenas dados de tamanho do projeto - como número de arquivos, linhas de código ou comentários, etc. Assim, as métricas que exploram maiores detalhes do código são poucas, como complexidade ciclomática.

4.1.4 C and C++ Code Counter

O C and C++ Code Counter (CCCC)[8] (Figura 4.4) é uma ferramenta de análise estática de código que inicialmente suportava apenas C e C++ mas hoje também suporta Java. A ferramenta não está em desenvolvimento ativo desde dezembro de 2011, mas como é código livre é possível que volte a ser desenvolvida mesmo sem a participação do desenvolvedor da última versão disponível.

A execução da ferramenta é por linha de comando, o que facilita a automatização da coleta de métricas. Os resultados são dados em HTML e XML, portanto fáceis de serem

[7] http://www.sonarsource.org
[8] http://cccc.sourceforge.net

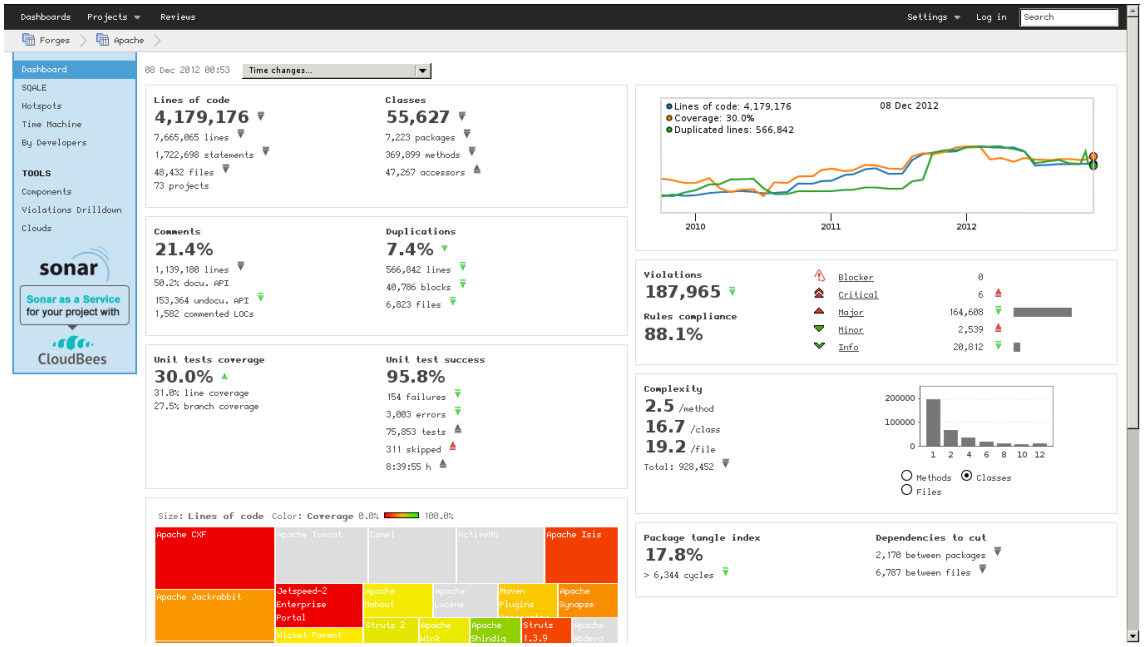


Figura 4.3: Sonar, com o código do Apache HTTP Server.

utilizados por outras ferramentas.

O CCCC não possui suporte a um grande número de métricas, apenas calculando 10 métricas. Algumas dessas são relevantes a Orientação a Objetos, como acoplamento entre objetos (CBO), enquanto outras são métricas tradicionais como número de linhas de código.

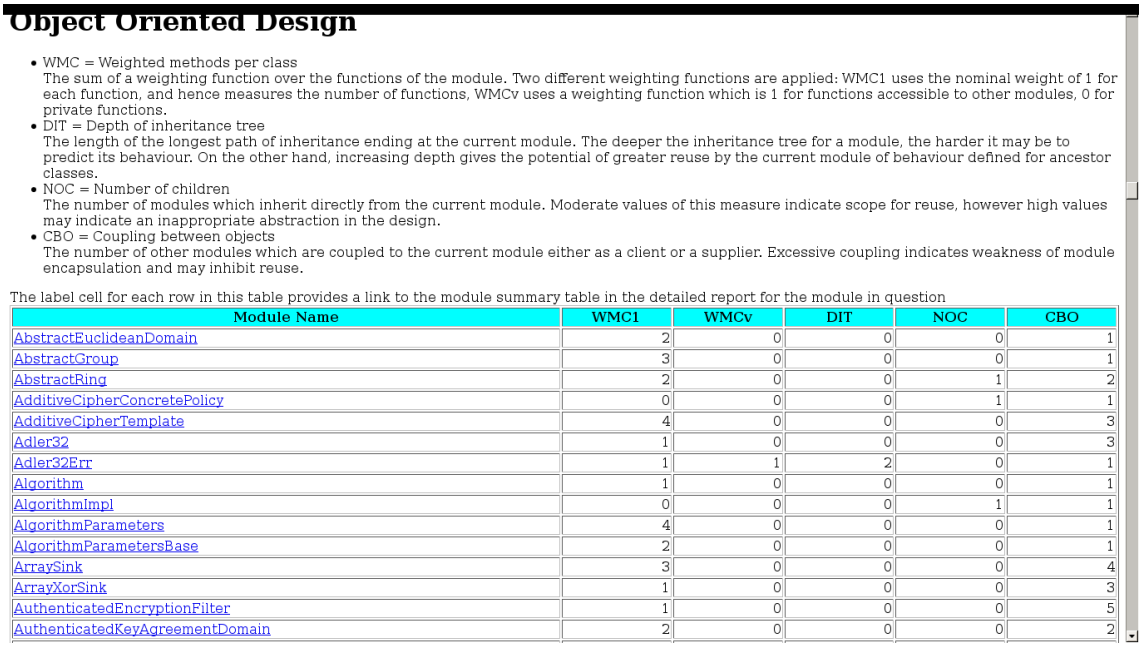
4.1.5 Understand

O Understand[9] (Figura 4.5) é uma ferramenta para análise estática de código fonte desenvolvido pela Scitools. A ferramenta suporta os seguintes sistemas operacionais: GNU/Linux, Mac OSX, Solaris e Windows. Além de suportar vários sistemas operacionais, também suporta várias linguagens: Ada, C, C++, C#, FORTRAN, Java, JOVIAL, Pascal, PL/M, VHDL, Cobol, PHP, HTML, CSS, Javascript, XML e Python. Apesar de somente necessitarmos suporte para C e C++, suporte a outras linguagens facilita trabalhos futuros que utilizem programas feitos em outras linguagens.

O Understand disponibiliza diversas formas de acesso aos dados que coleta, como programas de linha-de-comando (úteis para o desenvolvimento de /textitscripts para automatizar a coleta de métricas), exploração do código de forma visual, exportação de métricas para planilha e APIs em Python e Perl.

O Understand calcula várias das métricas comuns à maioria das ferramentas analisadas, como número de linhas de código e número de linhas de comentários. Além dessas, a ferramenta também disponibiliza 29 métricas de orientação a objetos, como profundidade máxima da árvore de herança (DIT), e 27 métricas de complexidade de código como complexidade ciclomática, que indica quão complexo o código é a partir do número de caminhos independentes.

[9] http://www.scitools.com



Object Oriented Design

- WMC = Weighted methods per class
  The sum of a weighting function over the functions of the module. Two different weighting functions are applied: WMC1 uses the nominal weight of 1 for each function, and hence measures the number of functions, WMCv uses a weighting function which is 1 for functions accessible to other modules, 0 for private functions.
- DIT = Depth of inheritance tree
  The length of the longest path of inheritance ending at the current module. The deeper the inheritance tree for a module, the harder it may be to predict its behaviour. On the other hand, increasing depth gives the potential of greater reuse by the current module of behaviour defined for ancestor classes.
- NOC = Number of children
  The number of modules which inherit directly from the current module. Moderate values of this measure indicate scope for reuse, however high values may indicate an inappropriate abstraction in the design.
- CBO = Coupling between objects
  The number of other modules which are coupled to the current module either as a client or a supplier. Excessive coupling indicates weakness of module encapsulation and may inhibit reuse.

The label cell for each row in this table provides a link to the module summary table in the detailed report for the module in question

Module Name	WMC1	WMCv	DIT	NOC	CBO
AbstractEuclideanDomain	2	0	0	0	1
AbstractGroup	3	0	0	0	1
AbstractRing	2	0	0	1	2
AdditiveCipherConcretePolicy	0	0	0	1	1
AdditiveCipherTemplate	4	0	0	0	3
Adler32	1	0	0	0	3
Adler32Err	1	1	2	0	1
Algorithm	1	0	0	0	1
AlgorithmImpl	0	0	0	1	1
AlgorithmParameters	4	0	0	0	1
AlgorithmParametersBase	2	0	0	0	1
ArraySink	3	0	0	0	4
ArrayXorSink	1	0	0	0	3
AuthenticatedEncryptionFilter	1	0	0	0	5
AuthenticatedKeyAgreementDomain	2	0	0	0	2

Figura 4.4: CCCC, com o código do Crypto++.

4.1.6 Ferramenta escolhida

Optou-se por utilizar o Understand devido ao amplo número de métricas que a ferramenta coleta (particularmente as métricas de orientação a objetos e complexidade), como pode ser visto na Tabela 4.1. As APIs disponibilizadas possibilitaram a automatização da coleta e seleção de métricas, para serem analisadas em outra ferramenta, contudo a inserção dos códigos-fonte na ferramenta foi feita manualmente. Como a ferramenta escolhida disponibiliza um número elevado de métricas não foi necessário o uso de várias ferramentas. Com isso evitou-se o problema de as ferramentas interpretarem as métricas de forma diferente (ou seja, para uma mesma métrica ferramentas diferentes produzem valores diferentes), demonstrado por (LINCKE; LUNDBERG; LÖWE, 2008).

Ferramenta	Número de métricas
Analizo	24
CCCC	10
Dependometer	17
Sonar	19
Understand	56

Tabela 4.1: Número de métricas calculadas pelas ferramentas analisadas.

4.2 Métricas Analisadas

Assim como nas outras ferramentas, diversas das métricas calculadas pelo extrator de métricas são redundantes - Complexidade Ciclomática, Complexidade Ciclomática Estrita e Complexidade Ciclomática Modificada, por exemplo, refletem as mesmas características. Outras, como número de linhas em branco ou número de linhas de comentários, não influem no código gerado. A análise dos dados foi limitada às métricas consideradas mais relevantes a quão bem orientado a objetos o código é.

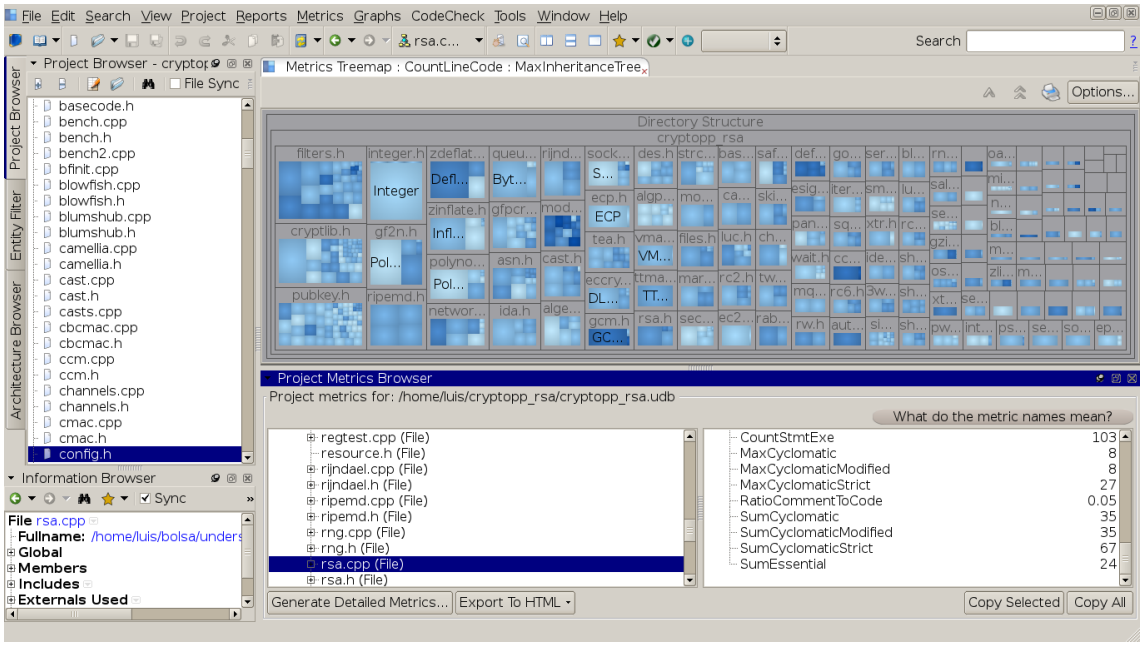


Figura 4.5: Understand, com o código do Crypto++.

4.2.1 Complexidade Ciclomática (V(G))

Introduzida por (MCCABE, 1976), a complexidade ciclomática é baseada no conceito de número ciclomático na teoria dos grafos. Associa-se ao programa um grafo de fluxo de controle, com um nodo de entrada e um nodo de saída. Para cada bloco do programa tem-se um nodo no grafo, e cada desvio corresponde a uma aresta. Um exemplo de grafo de fluxo de controle pode ser visto na Figura 4.6. A partir desse grafo, pode-se calcular a complexidade ciclomática da seguinte forma:

$$V(G) = e - n + 2p$$

onde e é o número de arestas, n é o número de nodos e p é o número de componentes conexos. Dessa forma, obtém-se o número de caminhos linearmente independentes entre o nodo de entrada e o nodo de saída.

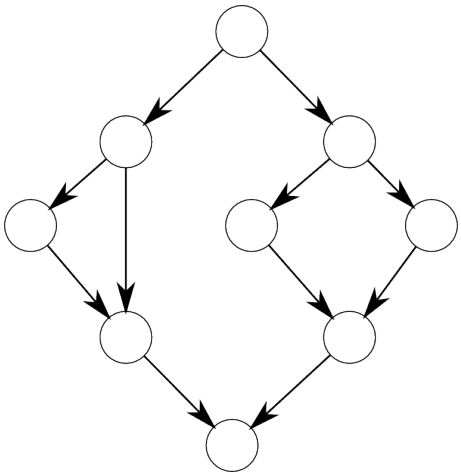


Figura 4.6: Exemplo de grafo de fluxo de controle.

4.2.2 Complexidade Ciclomática Máxima (WCM)

A complexidade ciclomática máxima de uma classe é o maior valor da complexidade ciclomática dos métodos aninhados.

4.2.3 Soma da Complexidade Ciclomática

A soma da complexidade ciclomática é definida como:

$$c_i = \text{Complexidade do i-ésimo método}$$

$$n = \text{Número de métodos da classe}$$

$$WCM = \sum_{i=1}^{n} c_i$$

4.2.4 Número de Classes Básicas Imediatas (IFANIN)

O número de classes básicas imediatas é o número de superclasses de uma classe. Assim, um número alto de classes básicas imediatas é um indicador de reúso de código e alta facilidade de manutenção.

4.2.5 Acoplamento Entre Classes de Objetos (CBO)

Uma das métricas introduzidas por (CHIDAMBER; KEMERER, 1994), o acoplamento entre classes de objetos, ou acoplamento aferente, é definido como o número de classes às quais uma classe está acoplada. De acordo com (CHIDAMBER; KEMERER, 1994), valores altos indicam maior dependência de uma classe de outras, o que dificulta o reúso e a manutenção do código, além de requerer maior número de testes.

4.2.6 Número de Filhos (NOC)

Assim como o acoplamento entre classes de objetos, o número de filhos também foi introduzido em (CHIDAMBER; KEMERER, 1994). O número de filhos é definido como o número de subclasses imediatas de uma classe. Visa a medir o número de classes que herdam da classe pai. Números maiores correspondem a maior reúso, mas números muito grandes indicam erros na criação das classes.

4.2.7 Fan-in e fan-out

Fan-in e fan-out são duas métricas estruturais introduzidas por (HENRY; KAFURA, 1981). Métricas estruturais têm como objetivo quantificar como os módulos relacionam-se entre si. Valores altos de fan-in e fan-out ocorrem devido a alto acoplamento, o que pode indicar que a função poderia ser dividida em mais etapas.

Fan-in Número de chamadas a uma função, acrescido do número de estruturas de dados lidas pela função.

Fan-out Número de chamadas feitas por uma função, acrescido do número de estruturas de dados atualizadas pela função.

4.2.8 Profundidade da Árvore de Herança (DIT)

Introduzida por (CHIDAMBER; KEMERER, 1994), a profundidade da árvore de herança de uma classe é a distância dessa classe até a raiz da árvore. Conforme uma classe

afasta-se da raiz aumenta o número de propriedades que podem ser herdadas e, portanto, maior é sua complexidade.

4.2.9 Aninhamento Máximo

O aninhamento máximo é o valor máximo para os níveis de aninhamento de construções de controle (como *if*, *for* e *while*) em uma função. Em geral, maiores valores indicam maior complexidade.

4.2.10 Percentual de Falta de Coesão entre Métodos

A Falta de Coesão entre Métodos (LCOM) foi introduzida por (CHIDAMBER; KEMERER, 1994) com o objetivo de mensurar quão coesa uma classe é através da semelhança entre seus métodos. Dois métodos A e B são considerados similares caso o conjunto das variáveis de instância utilizadas pelo método A possua pelo menos um elemento em comum com o conjunto das variáveis de instância utilizadas pelo método B.

$$\{I_i\} = \text{conjunto de variáveis de instância utilizadas pelo método } M_i$$

$$P = \{(I_i, I_j) | I_i \cap I_j = \emptyset\}$$

$$Q = \{(I_i, I_j) | I_i \cap I_j \neq \emptyset\}$$

$$LCOM = \begin{cases} |P| - |Q| & \text{se |P| > |Q|} \\ 0 & \text{caso contrário} \end{cases}$$

O Percentual de Falta de Coesão entre Métodos difere do LCOM por considerar não apenas se dois métodos possuem variáveis de instância em comum mas também o número dessas. Obtém-se o Percentual de Falta de Coesão entre Métodos da seguinte forma:

$$V = \text{Conjunto de variáveis de instância}$$

$$M_i = \text{Número de métodos que utilizam a variável de instância } i$$

$$t = \text{Número de métodos}$$

$$PercentLCOM = 1 - \frac{\sum_{i \in V} M_i}{|V| \cdot t}$$

Valores altos para Falta de Coesão indicam que a classe está fazendo tarefas que deveriam ser divididas em diversas classes. Uma classe que desempenha muitas funções tende a ser mais complexa e mais suscetível a erros que classes mais simples.

O Understand calcula as seguintes métricas em C/C++:

AltAvgLineBlank: Average Number of Blank Lines (Include Inactive) Média do número de linhas em branco para todos métodos ou funções aninhados, incluindo regiões inativas.

AltAvgLineCode: Average Number of Lines of Code (Include Inactive) Média do número de linhas contendo código fonte para todos métodos ou funções aninhados, incluindo regiões inativas.

AltAvgLineComment: Average Number of Lines with Comments (Include Inactive) Média do número de linhas contendo comentários para todos métodos ou funções aninhados, incluindo regiões inativas.

AltCountLineBlank: Blank Lines of Code (Include Inactive) Número de linhas em branco, incluindo regiões inativas.

AltCountLineCode: Lines of Code (Include Inactive) Número de linhas contendo código fonte, incluindo regiões inativas.

AltCountLineComment: Lines with Comments (Include Inactive) Número de linhas contendo comentários, incluindo regiões inativas.

AvgCyclomatic: Average Cyclomatic Complexity Média da complexidade ciclomática para todos métodos ou funções aninhados.

AvgCyclomaticModified: Average Modified Cyclomatic Complexity Média da complexidade ciclomática modificada para todos métodos ou funções aninhados.

AvgCyclomaticStrict: Average Strict Cyclomatic Complexity Média da complexidade ciclomática estrita para todos métodos ou funções aninhados.

AvgEssential: Average Essential Cyclomatic Complexity Média da complexidade essencial para todos métodos ou funções aninhados.

AvgLine: Average Number of Lines Média do número de linhas para todos métodos ou funções aninhados.

AvgLineBlank: Average Number of Blank Lines Média do número de linhas em branco para todos métodos ou funções aninhados.

AvgLineCode: Average Number of Lines of Code Média do número de linhas de código fonte para todos métodos ou funções aninhados.

AvgLineComment: Average Number of Lines with Comments Média do número de linhas contendo comentários para todos métodos ou funções aninhados.

CountClassBase: Base Classes Número de classes básicas imediatas.

CountClassCoupled: Coupling Between Objects Número de classes às quais uma classe está acoplada.

CountClassDerived: Number of Children Número de subclasses imediatas.

CountDeclClass: Classes Número de classes.

CountDeclClassMethod: Class Methods Número de métodos de uma classe.

CountDeclClassVariable: Class Variables Número de variáveis de uma classe.

CountDeclFunction: Function Número de funções.

CountDeclInstanceMethod: Instance Methods Número de métodos de instância.

CountDeclInstanceVariable: Instance Variables Número de variáveis de instância.

CountDeclInstanceVariablePrivate: Private Instance Variables Número de variáveis de instância privadas.

CountDeclInstanceVariableProtected: Protected Instance Variables Número de variáveis de instância protegidas.

CountDeclInstanceVariablePublic: Public Instance Variables Número de variáveis de instância públicas.

CountDeclMethod: Local Methods Número de métodos locais.

CountDeclMethodAll: Methods Número de métodos, incluindo métodos herdados.

CountDeclMethodConst: Local Const Methods Número de métodos constantes locais.

CountDeclMethodFriend: Friend Methods Número de métodos amigos locais.

CountDeclMethodPrivate: Private Methods Número de métodos privados.

CountDeclMethodProtected: Protected Methods Número de métodos protegidos.

CountDeclMethodPublic: Public Methods Número de métodos públicos.

CountInput: Inputs Número de parâmetros uma função usa mais o número de subprogramas que chamam a função (FANIN).

CountLine: Physical Lines Número de linhas.

CountLineBlank: Blank Lines of Code Número de linhas em branco.

CountLineCode: Source Lines of Code Número de linhas contendo código fonte.

CountLineCodeDecl: Declarative Lines of Code Número de linhas de código com declarações.

CountLineCodeExe: Executable Lines of Code Número de linhas com código fonte executável.

CountLineComment: Lines with Comments Número de linhas com comenarios.

CountLineInactive: Inactive Lines Número de linhas inativas.

CountLinePreprocessor: Preprocessor Lines Número de linhas de código para ser executado pelo preprocessador.

CountOutput: Outputs Número de subprogramas chamados mais número de variáveis globais (FANOUT).

CountPath: Paths Número de caminhos possíveis excluindo saídas anormais ou GOTOs (NPATH).

CountSemicolon: Semicolons Número de ponto e vírgulas.

CountStmt: Statements Número de statements.

CountStmtDecl: Declarative Statements Número de statements declarativos.

CountStmtEmpty: Empty Statements Número de statements vazios

CountStmtExe: Executable Statements Número de statements executáveis

Cyclomatic: Cyclomatic Complexity Complexidade ciclomática.

CyclomaticModified: Modified Cyclomatic Complexity Similar a complexidade ciclomática, mas decisões em switch/case contam, ao todo, como uma única decisão.

CyclomaticStrict: Strict Cyclomatic Complexity Similar a complexidade ciclomática, porém cada conjunção lógica (e, ou) adiciona 1 na complexidade estrita.

Essential: Essential Complexity Similar a complexidade ciclomática, mas estruturas de controle como if-then-else e loops são consideradas como um único statement.

MaxCyclomatic: Max Cyclomatic Complexity Complexidade ciclomática máxima das funções e métodos aninhados.

MaxCyclomaticModified: Max Modified Cyclomatic Complexity Complexidade ciclomática modificada máxima das funções e métodos aninhados.

MaxCyclomaticStrict: Max Strict Cyclomatic Complexity Complexidade ciclomática estrita máxima das funções e métodos aninhados.

Max Essential Complexity Complexidade ciclomática essencial máxima das funções e métodos aninhados.

MaxEssentialKnots: Max Knots Máximo de nós após remover construções de programação estruturada.

MaxInheritanceTree: Depth of Inheritance Tree A profundidade na árvore de herança é dada pelo máximo número de nodos da classe até a raíz da árvore de herança.

MaxNesting: Nesting Profundidade máxima de construções de controle (if, while, etc).

MinEssentialKnots: Minimum Knots Mínimo de nós após remover construções de programação estruturada.

PercentLackOfCohesion: Lack of Cohesion in Methods 100 % menos a coesão média dos membros da classe.

RatioCommentToCode: Comment to Code Ratio Proporção de comentários para código.

SumCyclomatic: Sum Cyclomatic Complexity Soma das complexidades ciclomáticas de todas funções ou métodos aninhados.

SumCyclomaticModified: Sum Modified Cyclomatic Complexity Soma das complexidades ciclomáticas modificadas de todas funções ou métodos aninhados.

SumCyclomaticStrict: Sum Strict Cyclomatic Complexity Soma das complexidades ciclomáticas estritas de todas funções ou métodos aninhados.

SumCyclomaticEssential: Sum Essential Complexity Soma das complexidades ciclomáticas essenciais de todas funções ou métodos aninhados.

Algumas das métricas são calculadas para cada função ou método individualmente. Para poder comparar os diferentes programas, essas métricas foram agrupadas da seguinte forma:

Cada uma das métricas a seguir foi agrupada somando-se seus respectivos valores:

AltCountLineBlank	AltCountLineCode
AltCountLineComment	CountClassBase
CountClassCoupled	CountClassDerived
CountDeclClass	CountDeclClassMethod
CountDeclClassVariable	CountDeclFunction
CountDeclInstanceMethod	CountDeclInstanceVariable
CountDeclInstanceVariablePrivate	CountDeclInstanceVariableProtected
CountDeclInstanceVariablePublic	CountDeclMethod
CountDeclMethodAll	CountDeclMethodConst
CountDeclMethodFriend	CountDeclMethodPrivate
CountDeclMethodProtected	CountDeclMethodPublic
CountInput	CountLine
CountLineBlank	CountLineCode
CountLineCodeDecl	CountLineCodeExe
CountLineComment	CountLineInactive
CountLinePreprocessor	CountOutput
CountPath	CountSemicolon
CountStmt	CountStmtDecl
CountStmtEmpty	CountStmtExe
Cyclomatic	CyclomaticModified
CyclomaticStrict	Essential
Knots	SumCyclomatic
SumCyclomaticModified	SumCyclomaticStrict
SumEssential	

Cada uma das métricas a seguir foi agrupada tomando-se a média de seus respectivos valores:

AltAvgLineBlank	AltAvgLineCode
AltAvgLineComment	AvgCyclomatic
AvgCyclomaticModified	AvgCyclomaticStrict
AvgEssential	AvgLine
AvgLineBlank	AvgLineCode
AvgLineComment	PercentLackOfCohesion
RatioCommentToCode	

Para MinEssentialKnots utilizou-se o mínimo dos valores.

Cada uma das métricas a seguir foi agrupada escolhendo-se o valor máximo dentre seus respectivos valores:

MaxCyclomatic	MaxCyclomaticModified
MaxCyclomaticStrict	MaxEssentialKnots
MaxInheritanceTree	MaxNesting

4.3 Filtragem

Alguns dos aplicativos utilizados possuem grande porção de código que não é executado durante o *benchmark*. Por exemplo, o Crypto++ implementa vários algoritmos que não foram utilizados em nenhum *benchmark*. Contudo, a ferramenta de extração de métricas utilizada não exclui essa porção de código do cálculo das métricas, resultando em métricas que não refletem o código executado tão bem quanto poderiam.

Para melhorar a qualidade das métricas coletadas é necessário saber quais partes do código foram executadas. Essa informação pode ser obtida fazendo-se uma análise dinâmica do programa, por meio de um *profiler*, ou seja, executando o programa e verificando quais trechos são executados.

4.3.1 Profiling

Um *profiler* é um programa que faz análise dinâmica de um aplicativo, ou seja, analisa a aplicação durante a sua execução. Isso permite obter várias informações indisponíveis apenas com a análise do código fonte, como quais métodos são utilizados quando o programa é chamado como uma dada entrada ou quanto de memória o aplicativo utilizou.

4.3.1.1 gprof

Inicialmente a análise da execução foi feita utilizando o gprof, que é uma ferramenta de *profiling* bastante conhecida e fácil de usar. O gprof funciona inserindo código, durante a compilação, em todas funções do programa. Quando o programa é executado o código inserido coleta diversas estatísticas. O gprof disponibiliza os nomes das funções executadas, bem como quantas vezes cada função foi chamada e quanto tempo levou para executar. Contudo, como C++ permite sobrecarga de métodos (isto é, o nome do método não necessariamente identifica-o univocamente), é difícil mapear a saída do gprof com o código do programa.

4.3.1.2 Valgrind

Como não foi possível obter as informações desejadas do gprof, tentou-se outro *profiler*, o Valgrind. Enquanto o gprof é apenas uma ferramenta para *profiling*, o Valgrind executa o aplicativo a ser mensurado em uma máquina virtual (SIMPSON; MIDDHA; BARUA, 2005). Isso resulta em certo *overhead*, mas todos os programas mensurados terminaram em poucos minutos - ou seja, o *overhead* não foi proibitivamente alto nesses casos.

A grande vantagem do Valgrind sobre o gprof é disponibilizar um *framework* que permite o desenvolvimento de ferramentas para controle e coleta de dados da execução. O Valgrind possui diversas ferramentas oficiais, e uma delas, Callgrind, coleta dados do código executado, mas, como a ferramenta coleta diversos outros dados de execução, neste trabalho optou-se por desenvolver uma ferramenta que apenas lista o nome da função, arquivo de origem, linha, e endereço no binário.

4.3.2 Filtragem por Arquivo

Unindo os dados do *profiler* com a API do extrator de métricas, pode-se filtrar grande parte do código não utilizado da coleta de métricas. Contudo, não foi filtrado todo código não executado devido a isso ser bastante complexo em muitas linguagens de programação, C e C++ inclusive. Optou-se pela solução adotada por (CORRÊA, 2011), de considerar para o cálculo das métricas todo o conteúdo dos arquivos com uma ou mais funções

executadas.

4.4 Resultados

Os resultados das métricas coletadas, após a filtragem, podem ser vistos nas tabelas 4.2, 4.3, 4.4 e 4.5. Para diversos dos aplicativos mensurados as métricas CountClassBase, CountClassDerived, MaxInheritanceTree e PercentLackOfCohesion obtiveram o valor zero. Isso deve-se a serem métricas que envolvem atributos não presentes em aplicativos escritos na linguagem C, como classes. Em outros casos, como para a métrica CountClassCoupled, mesmo essa sendo definida para orientação a objetos, os aplicativos em C possuem valores não nulos devido ao fato de a ferramenta utilizada para obter as métricas considerar construções que não são classes, como *struct* no caso da linguagem C, como sendo equivalentes a classes para a contagem da métrica (LINCKE; LUNDBERG; LÖWE, 2008).

nome	AvgCyclomatic	CountClassBase	CountClassCoupled
cjpeg	2.17647058824	0	19
crypto++ aes	3.73356807512	601	1455
crypto++ des	3.73443008226	601	1455
crypto++ rsa	3.65458663647	613	1484
crypto++ sign/verify	3.67046750285	611	1482
djpeg	2.62790697674	0	20
gzip	3.72727272727	0	5
h263dec	3.61904761905	0	0
h263enc	4.15384615385	0	1
h264dec	3.23456790123	0	18
h264enc	2.88695652174	0	29
jpg2000dec	0.789215686275	0	92
jpg2000enc	0.837438423645	0	116
links	1.12571428571	0	115
mibench blowfish	3.875	0	0
mibench rijndael	1496.0	0	0
mpeg1	2.03846153846	0	12
mpeg2dec	5.73684210526	0	0
mpeg2dec++	3.47142857143	4	84
mpeg2enc	8.83333333333	0	0
sablotron	3.44876033058	450	4211
sha	1.5	0	0
xerces	3.31452267908	439.0	3936.5

Tabela 4.2: Métricas analisadas.

nome	CountClassDerived	CountInput	CountOutput
cjpeg	0	1244	885
crypto++ aes	568	6010	5223
crypto++ des	568	5990	5202
crypto++ rsa	571	6418	5763
crypto++ sign/verify	569	6354	5709
djpeg	0	1324	1006
gzip	0	690	535
h263dec	0	576	300
h263enc	0	852	502
h264dec	0	3791	2227
h264enc	0	7395	3556
jpg2000dec	0	2936	2126
jpg2000enc	0	2855	1999
links	0	9929	6476
mibench blowfish	0	32	16
mibench rijndael	0	39	28
mpeg1	0	1881	1212
mpeg2dec	0	949	645
mpeg2dec++	4	1260	922
mpeg2enc	0	968	421
sablotron	508	31195	23354
sha	0	29	23
xerces	471.5	26899	20443

Tabela 4.3: Métricas analisadas.

nome	MaxCyclomatic	MaxInheritanceTree	MaxNesting
cjpeg	53	0	7
crypto++ aes	117	8	5
crypto++ des	117	8	5
crypto++ rsa	117	8	5
crypto++ sign/verify	117	8	5
djpeg	3	0	7
gzip	56	0	6
h263dec	117	0	7
h263enc	64	0	6
h264dec	325	0	12
h264enc	175	0	11
jpg2000dec	39	0	8
jpg2000enc	64	0	10
links	75	0	8
mibench blowfish	14	0	3
mibench rijndael	11203	0	2
mpeg1	68	0	7
mpeg2dec	38	0	5
mpeg2dec++	34	1	5
mpeg2enc	64	0	7
sablotron	147	5	10
sha	7	0	3
xerces	111	4	10

Tabela 4.4: Métricas analisadas.

nome	PercentLackOfCohesion	SumCyclomatic
cjpeg	0.0	787
crypto++ aes	18.1820768137	7575
crypto++ des	18.1820768137	7558
crypto++ rsa	17.8675862069	7892
crypto++ sign/verify	17.8476454294	7859
djpeg	0.0	884
gzip	0.0	670
h263dec	0.0	620
h263enc	0.0	903
h264dec	0.0	3661
h264enc	0.0	5785
jpg2000dec	0.0	2572
jpg2000enc	0.0	2572
links	0.0	6116
mibench blowfish	0.0	31
mibench rijndael	0.0	22448
mpeg1	0.0	1722
mpeg2dec	0.0	809
mpeg2dec++	26.3461538462	1720
mpeg2enc	0.0	903
sablotron	40.8391376451	34595
sha	0.0	21
xerces	48.3373553271	30038.5

Tabela 4.5: Métricas analisadas.

5 SIMULADOR

Um simulador é uma ferramenta que simula vários componentes de hardware e suas interações. Existem diferentes níveis de simulação (UBAL et al., 2011):

Simulação Funcional Centrada na simulação da funcionalidade dos componentes, mas não simula os detalhes do hardware não necessários do ponto de vista da aplicação simulada, como, por exemplo, diversos níveis de cache (MARTIN et al., 2005);

Simulação Arquitetural Modela estruturas de hardware como *pipelines*, filas de instruções, unidades funcionais e *caches*, considerando características físicas como atrasos, mas sem simular o circuito em si (UBAL et al., 2007).

Utilizou-se simulação arquitetural, de forma a obter maior controle da simulação e mais informações da execução. O simulador utilizado para obtenção das métricas físicas foi o Multi2Sim (UBAL et al., 2012), que simula a execução de programas x86 no sistema operacional GNU/Linux a nível da aplicação, ou seja, chamadas de sistema não são simuladas. Todas chamadas de sistema feitas pelo programa simulado são executadas no sistema operacional da máquina em que a simulação está executando. Isso resulta em simulações mais rápidas e menos dependentes de como as chamadas de sistema foram implementadas, contudo os resultados podem ser enviesados caso a aplicação simulada seja muito dependente de chamadas de sistema.

A *Instruction Set Architecture* (ISA) simulada pelo Multi2Sim, Intel x86, é uma das ISAs mais usadas, principalmente em computadores pessoais. Em sistemas embarcados, a presença dessa ISA é limitada, mas está presente por meio dos seguintes processadores: Atom (Intel), Athlon Neo (AMD) e Nano (VIA). Uma das vantagens que a arquitetura apresenta é o amplo número de softwares escritos para ela e que não estão disponíveis em outras arquiteturas como ARM ou SPARC. As razões pela escolha dessa arquitetura e não outras mais comuns a sistemas embarcados foram ampla disponibilidade de software, facilidade de compilar os aplicativos a serem simulados e facilidade de simulação.

O simulador também permite simular a execução de programas OpenCL na GPU. Esse aspecto do simulador não foi utilizado devido aos aplicativos selecionados executarem somente na CPU.

5.1 Configurações

O Multi2Sim permite simular diferentes configurações da arquitetura x86. Pode-se controlar características como predição de desvios, detalhes de pipeline, cache e número de unidades funcionais. Com o objetivo de simular processadores com diferentes pipelines, ou seja, com maior ou menor capacidade de executar as instruções paralelamente,

foram utilizadas duas diferentes configurações, como pode ser visto na Tabela 5.1. Os outros valores configuráveis utilizados foram mantidos constantes entre as configurações, e podem ser vistos nas tabelas 5.2, 5.3, 5.4, 5.5 e 5.6.

	Descrição	config. 1	config. 2
DecodeWidth	Número de instruções decodificadas por ciclo	1	8
DispatchWidth	Número de microinstruções *dispatched* por ciclo	1	8
IssueWidth	Número de microinstruções *issued* por ciclo	1	8
CommitWidth	Número de microinstruções *committed* por ciclo	1	8

Tabela 5.1: Configurações utilizadas para calcular o Δ IPC.

Nome	Descrição	Valor
FetchQueueSize	Tamanho da *fetch queue*	64 bytes
UopQueueSize	Tamanho da *uop queue*	32
RfIntSize	Número de registradores inteiros físicos por *thread*	80
RfFpSize	Número de registradores ponto-flutuante físicos por *thread*	40

Tabela 5.2: Características de fila comuns a todas configurações.

Nome	Descrição	Valor
Kind	Tipo do preditor de desvios	TwoLevel
TwoLevel.L1Size	Tamanho do primeiro nível	1
TwoLevel.L2Size	Tamanho do segundo nível	1024
TwoLevel.HistorySize	Tamanho do histórico de desvio do segundo nível	8

Tabela 5.3: Características do preditor de desvios comuns a todas configurações.

Nome	Descrição	Valor para L1	Valor para L2
Sets	Número de conjuntos na cache	256	1024
Assoc	Associatividade	2	8
BlockSize	Tamanho em bytes de um bloco	64	64
Latency	Latência em número de ciclos	2	20
Policy	Política de substituição	LRU	LRU
ReadPorts	Número de portas de leitura	2	2
WritePorts	Número de portas de escrita	1	1

Tabela 5.4: Características gerais de cache comuns a todas configurações.

Nome	Descrição	Valor
Cores	Número de *cores*	1
Threads	Número de hardware *threads* por *core*	1
RecoverKind	Quando ocorre erro de desvio, é o estágio da execução do desvio errado em que a recuperação inicia	Writeback
RecoverPenalty	Número de ciclos que o *fetch stage* (estágio de busca) fica parado ao errar uma predição de desvio	0
PageSize	Tamanho da página de memória	4kB

Tabela 5.5: Características gerais comuns a todas configurações.

Nome	Descrição	Valor
IntAdd.Count	Número de somadores de inteiros	4
IntAdd.OpLat	*Operation latency* para soma de inteiros	2
IntAdd.IssueLat	*Issue latency* para soma de inteiros	1
IntSub.Count	Número de subtratores de inteiros	4
IntSub.OpLat	*Operation latency* para subtração de inteiros	2
IntSub.IssueLat	*Issue latency* para o subtração de inteiros	1
IntMult.Count	Número de multiplicadores de inteiros	1
IntMult.OpLat	*Operation latency* para multiplicação de inteiros	3
IntMult.IssueLat	*Issue latency* para multiplicação de inteiros	3
IntDiv.Count	Número de divisores de inteiros	1
IntDiv.OpLat	*Operation latency* para divisão de inteiros	20
IntDiv.IssueLat	*Issue latency* para divisão de inteiros	20
EffAddr.Count	Número de operadores de cálculo de endereço efetivo	4
EffAddr.OpLat	*Operation latency* para cálculo de endereço efetivo	2
EffAddr.IssueLat	*Issue latency* para cálculo de endereço efetivo	1
Logical.Count	Número de operadores para operações lógicas	4
Logical.OpLat	*Operation latency* para operações lógicas	1
Logical.IssueLat	*Issue latency* para operações lógicas	1
FpSimple.Count	Número de operadores operações simples de ponto-flutuante	2
FpSimple.OpLat	*Operation latency* para operações simples de ponto-flutuante	2
FpSimple.IssueLat	*Issue latency* para operações simples de ponto-flutuante	2
FpAdd.Count	Número de somadores ponto-flutuante	2
FpAdd.OpLat	*Operation latency* para soma em ponto-flutuante	5
FpAdd.IssueLat	*Issue latency* para soma em ponto-flutuante	5
FpMult.Count	Número de multiplicadores ponto-flutuante	1
FpMult.OpLat	*Issue latency* para multiplicação em ponto-flutuante	10
FpMult.IssueLat	*Issue latency* para multiplicação em ponto-flutuante	10
FpDiv.Count	Número de divisores ponto-flutuante	1
FpDiv.OpLat	*Issue latency* para divisão em ponto-flutuante	20
FpDiv.IssueLat	*Issue latency* para divisão em ponto-flutuante	20
FpComp.Count	Número de comparadores ponto-flutuante	2
FpComp.OpLat	*Issue latency* para comparação em ponto-flutuante	5
FpComp.IssueLat	*Issue latency* para comparação em ponto-flutuante	5
FpComplex.Count	Número de operadores para operações complexas em ponto-flutuante	1
FpComplex.OpLat	*Issue latency* para operações complexas em ponto-flutuante	40
FpComplex.IssueLat	*Issue latency* para operações complexas em ponto-flutuante	40

Tabela 5.6: Características das unidades funcionais comuns a todas configurações.

5.2 Adições ao Simulador

Como o simulador ainda não implementa todas instruções da arquitetura, foi necessário implementar algumas instruções utilizadas pelos *benchmarks*, mesmo compilando esses sem otimizações. Implementou-se as instruções a seguir, seguindo a descrição em (INTEL, 2012) e elas foram incorporadas na versão oficial do simulador:

REP STOSW Move uma palavra do registrador AX para a posição de memória apontada por ES:DI. Incrementa o registrador DI se a DF (*Direction Flag*) for 0, ou, se DF

for 1, o registrador DI é decrementado. Como possui o prefixo REP, o registrador CX é decrementado e o processo repetido até CX ser zero;

POP ESP Incrementa o registrador ESP (*Extended Stack Pointer*) e move o valor no topo da pilha para o registrador ESP.

5.3 Métricas Físicas

O simulador disponibiliza métricas físicas como instruções executadas, IPC (Instructions Per Cycle), cache misses, número de acessos à memória, memória utilizada e tempo de execução. Métricas como consumo de potência não são calculadas pelo simulador. Para possibilitar a comparação de programas com diferentes tempos de execução, decidiu-se utilizar a variação de IPC entre duas diferentes configurações, com diferentes capacidades de pipeline. Quanto maior a variação de IPC, melhor utilizado é o pipeline.

5.4 Resultados

Os valores de instruções por ciclo obtidos nas simulações podem ser vistos na Tabela 5.7:

Benchmark	Δ IPC	IPC da configuração 1	IPC da configuração 2
aes	1.603944	0.963915	2.567859
des	1.612143	0.945637	2.557780
rsa	0.812475	0.956890	1.769365
sign/verify	1.330979	0.975108	2.306087
mibench blowfish	1.793045	0.993551	2.786596
mibench rijndael	2.121686	0.983642	3.105328
sha	0.826239	0.866638	1.692877
cjpeg	1.717676	0.953123	2.670799
djpeg	1.458840	0.978194	2.437034
jpeg2000dec	0.801317	0.793443	1.594760
jpeg2000enc	0.796449	0.796663	1.593112
h263dec	1.195832	0.917170	2.113002
h263enc	1.893411	0.980355	2.873766
h264dec	1.065655	0.870729	1.936384
h264enc	1.150994	0.881014	2.032008
mpeg1	1.256194	0.946800	2.202994
mpeg2dec	1.176904	0.925994	2.102898
mpeg2enc	2.111496	0.946473	3.057969
gzip	0.511236	0.764949	1.276185
links	1.105356	0.845300	1.950656
sablot	0.838217	0.832854	1.671071
xerces	1.373526	0.903537	2.277063

Tabela 5.7: Resultados das simulações.

6 ANÁLISE DE CORRELAÇÃO

Para a análise dos dados coletados foram considerados diferentes métodos estatísticos. Segue uma breve descrição de cada método, e, após, os resultados obtidos.

6.1 Método dos Mínimos Quadrados

Atribuído a Carl Friedrich Gauss, 1795, mas publicado pela primeira vez por Adrien Marie Legendre em 1805 no livro *Nouvelles methods pour la determination des orbites des cometes*, o método de mínimos quadrados é, ainda hoje, a técnica estatística moderna não trivial mais utilizada, segundo (STIGLER, 1986).

Segundo (WOLBERG; WOLBERG, 2006), no caso mais simples, conhecido como regressão linear, o método utiliza apenas uma variável independente (x), e uma variável dependente (y). O caso geral, que utiliza várias variáveis independentes, é chamado de regressão linear múltipla.

Neste trabalho, as variáveis independentes são as métricas de qualidade de software e as variáveis dependentes são as métricas físicas. Ou seja, o tipo de regressão utilizada é regressão linear múltipla.

Dadas as variáveis independentes $x_{1...n,1...p}$, as variáveis dependentes $y_{1...n}$, os pesos $\beta_{1...p}$ e os resíduos (erros) ε_i, y pode ser aproximado por uma função $f(x_i)$ da seguinte forma:

$$y_i = f(x_i) + \varepsilon_i$$
$$f(x_i) = \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$$

O método dos mínimos quadrados consiste em minimizar a equação a seguir, chamada de função objetivo:

$$S = \sum_{i=1}^{n} w_i (y_i - f(x_i))^2$$

Os pesos (w_i) da função objetivo advêm de incertezas nas medições, ou seja, quão maiores forem as incertezas presentes em um ponto, menor o seu peso. Como neste trabalho não foram utilizados instrumentos de diferentes precisões nas mensurações, utilizou-se o valor 1 para todos pesos.

O método de mínimos quadrados foi utilizado por meio da função regress do Matlab. Devido ao número de variáveis independentes ser grande comparado ao tamanho da amostra, ao invés de fazer uma única regressão com todas variáveis fizeram-se múltiplas regressões, utilizando-se todas as combinações das variáveis em grupos de uma, duas e três variáveis.

Uma forma de mensurar quão bem o método dos mínimos quadrados aproxima o resultado é através do coeficiente de determinação (R^2). De acordo com (EVERITT, 2002), o coeficiente de determinação dá a percentagem da variação de uma variável que pode ser explicada pela outra, ou seja, quanto maior o R^2, melhor é a qualidade da predição. O R^2 é o quadrado do coeficiente de correlação e pode ser obtido da seguinte forma:

$$SSE = \sum_{i}^{n} (y_i - f(x_i))^2$$

$$\overline{y} = \frac{\sum_{n}^{i} y}{n}$$

$$SS_{yy} = \sum_{i}^{n} (y_i - \overline{y})^2$$

$$R^2 = \frac{SS_{yy} - SSE}{SS_{yy}} = \frac{SS_{yy}}{SS_{yy}} - \frac{SSE}{SS_{yy}} = 1 - \frac{SSE}{SS_{yy}}$$

Os 15 melhores resultados obtidos no Matlab, utilizando o R^2 como base, para cada grupo podem ser vistos nas tabelas 6.1, 6.2 e 6.3 (no total foram 11 regressões para uma variável, 55 para duas e 165 para três). Além do R^2, as tabelas mostram o valor do erro médio quadrático (MSE) para cada regressão, que pode ser obtido dividindo-se o SSE pelo número de amostras.

métrica	R^2	MSE
MaxNesting	0.223561	0.148710
CountInput	0.186121	0.155880
CountOutput	0.180650	0.156928
AvgCyclomatic	0.170054	0.158958
MaxCyclomatic	0.164404	0.160040
CountClassCoupled	0.133788	0.165904
PercentLackOfCohesion	0.112593	0.169963
SumCyclomatic	0.058535	0.180317
CountClassDerived	0.039287	0.184003
CountClassBase	0.028800	0.186012
MaxInheritanceTree	0.011029	0.189416

Tabela 6.1: Resultados da regressão para grupos com uma métrica de qualidade.

métrica 1	métrica 2	R^2	MSE
CountInput	SumCyclomatic	0.318829	0.130463
CountOutput	SumCyclomatic	0.312739	0.131630
AvgCyclomatic	CountInput	0.310496	0.132059
CountInput	MaxCyclomatic	0.308156	0.132507
AvgCyclomatic	CountOutput	0.306388	0.132846
CountOutput	MaxCyclomatic	0.303835	0.133335
AvgCyclomatic	SumCyclomatic	0.301647	0.133754
MaxCyclomatic	SumCyclomatic	0.297964	0.134459
AvgCyclomatic	MaxNesting	0.285231	0.136898
MaxCyclomatic	MaxNesting	0.285133	0.136917
AvgCyclomatic	CountClassCoupled	0.270209	0.139775
CountClassCoupled	MaxCyclomatic	0.266895	0.140410
CountOutput	MaxNesting	0.266250	0.140534
CountInput	MaxNesting	0.262412	0.141269
CountClassCoupled	MaxNesting	0.261079	0.141524

Tabela 6.2: Resultados da regressão para grupos com duas métricas de qualidade.

métrica 1	métrica 2	métrica 3	R^2	MSE
CountOutput	MaxNesting	SumCyclomatic	0.343270	0.125782
AvgCyclomatic	CountClassCoupled	CountOutput	0.341817	0.126060
CountClassCoupled	CountOutput	MaxCyclomatic	0.339980	0.126412
CountClassCoupled	CountOutput	SumCyclomatic	0.339508	0.126503
CountClassDerived	CountOutput	SumCyclomatic	0.337773	0.126835
AvgCyclomatic	CountOutput	MaxNesting	0.336463	0.127086
CountClassBase	CountOutput	SumCyclomatic	0.336275	0.127122
CountOutput	MaxCyclomatic	MaxNesting	0.336179	0.127140
CountOutput	MaxInheritanceTree	SumCyclomatic	0.336158	0.127144
AvgCyclomatic	CountClassCoupled	SumCyclomatic	0.335492	0.127272
AvgCyclomatic	CountInput	MaxNesting	0.334685	0.127426
CountInput	MaxCyclomatic	MaxNesting	0.334351	0.127490
CountInput	MaxNesting	SumCyclomatic	0.333402	0.127672
AvgCyclomatic	MaxNesting	SumCyclomatic	0.332796	0.127788
MaxCyclomatic	MaxNesting	SumCyclomatic	0.332434	0.127857

Tabela 6.3: Resultados da regressão para grupos com três métricas de qualidade.

Como pode ser visto nas tabelas 6.1, 6.2, 6.3, R^2 teve valores bastante baixos, mesmo nos melhores casos. Isso indica que as variáveis independentes selecionadas não são boas preditoras para o ΔIPC do conjunto de aplicações selecionado.

Assim, decidiu-se verificar se existe correlação entre alguma das 66 métricas de qualidade e o ΔIPC, mesmo as que inicialmente não foram consideradas. A combinação dessas variáveis em grupos de uma, duas e três variáveis resulta em um número alto de regressões (66, 2145 e 45760, respectivamente). Isso significa que existe um alto número de hipóteses a serem testadas, cada uma com uma chance individual (baixa) de erro. Caso os testes de hipótese sejam feitos sem controle algum, devido ao alto número de testes, a chance de erro é extremamente alta. Assim, é necessário o uso de métodos estatísticos

que garantam que o resultado não advenha apenas da soma dos erros de cada uma das muitas hipóteses testadas. O método utilizado para obter o controle do erro é mostrado a seguir.

6.2 Taxa de Falsas Descobertas (FDR)

A taxa de falsas descobertas é a proporção esperada de erros do tipo I (isto é, a proporção de rejeições da hipótese nula quando esta é verdadeira). As hipóteses nulas utilizadas neste trabalho são da seguinte forma: "as métricas A, B e C não estão correlacionadas ao ΔIPC", o que significa que ao rejeitarmos uma dessas hipóteses afirmamos a existência de correlação. O controle do número de erros do tipo I é bastante útil quando múltiplos testes de hipótese são feitos - caso não seja feito algum controle, um número suficientemente grande de testes de hipótese garante a rejeição errônea de diversas das hipóteses.

Uma forma de limitar a taxa de falsas descobertas é através do método proposto por (BENJAMINI; HOCHBERG, 1995). O método é baseado na correção de Bonferroni e permite controlar a proporção de falsas descobertas, mas, ao mesmo tempo, reduz a potência estatística, ou seja, aumenta a chance de se cometerem erros do tipo II (aceitar uma hipótese que deveria ser rejeitada). A taxa de falsas descobertas é definida como:

S Número de hipóteses corretamente rejeitadas

V Número de hipóteses erroneamente rejeitadas

R Número de hipóteses rejeitadas, ou seja, $V + S$

Q Taxa de falsas descobertas = $\frac{V}{R}$ caso $R > 0$, ou 0 caso contrário

$$taxa\ de\ falsas\ descobertas\ (FDR) = E(Q) = E(\frac{V}{V+S}) = E(\frac{V}{R})$$

Ou seja, FDR é a proporção de falsos positivos para o total de hipóteses rejeitadas. Como os valores de S, V e Q são desconhecidos, o procedimento de BH propõe controlar a taxa de falsas descobertas da seguinte forma. Em (REINER; YEKUTIELI; BENJAMINI, 2003), o procedimento é descrito da seguinte forma:

Dadas m hipóteses, m_0 das quais verdadeiras, para cada hipótese H_i calcula-se o p-value correspondente P_i. Ordena-se os valores em ordem crescente de acordo com o p-value (ou seja, H_i corresponde a hipótese com o i-ésimo menor p-value). Para um nível de FDR arbitrário q, os p-values P_i são comparados ao valor crítico $q \cdot \frac{i}{m}$. Sendo $k = max\{i : P_i \leq q \cdot \frac{i}{m}\}$, rejeita-se $H_1, \ldots, H_k$ se existir k.

Segundo (BENJAMINI; HOCHBERG, 1995), esse procedimento garante que $FDR \leq q$, ou seja, permite controlar a taxa de falsas descobertas a partir do q.

Escolheu-se diversos valores para q, começando em 0.01 e incrementando esse valor em passos de 0.01 até que pelo menos um resultado fosse aceito. Cinco dos melhores resultados, ordenados por menor R^2, bem como o limitante superior q para as probabilidades de falsas descobertas correspondentes, podem ser vistos abaixo:

AvgCyclomatic R^2: 0.170054, MSE: 0.158958, q: 0.13

CountLineComment R^2: 0.169937, MSE: 0.158980, q: 0.13

Knots R^2: 0.165117, MSE: 0.159903, q: 0.13

MaxCyclomatic R^2: 0.164404, MSE: 0.160040, q: 0.13

AvgLineCode R^2: 0.133441, MSE: 0.165970, q: 0.13

CountDeclMethodProtected , CountSemicolon R^2: 0.243955, MSE: 0.144804, q: 0.15

SumEssential , SumCyclomatic R^2: 0.243946, MSE: 0.144805, q: 0.15

RatioCommentToCode , MaxNesting R^2: 0.225144, MSE: 0.148406, q: 0.15

CountLineCode , CountLineCodeExe R^2: 0.225097, MSE: 0.148416, q: 0.15

CountStmt , CountLineCodeExe R^2: 0.225081, MSE: 0.148419, q: 0.15

AvgLineComment , MaxNesting , AvgCyclomaticModified R^2: 0.316791, MSE: 0.130854, q: 0.14

SumCyclomatic , CountLine , CountLineComment R^2: 0.316786, MSE: 0.130854, q: 0.14

CountClassDerived , MaxCyclomatic , CountLineBlank R^2: 0.316783, MSE: 0.130855, q: 0.14

AltCountLineComment , CyclomaticStrict , CountLineCodeDecl R^2: 0.316757, MSE: 0.130860, q: 0.14

CountLineCode , MaxCyclomaticModified , AvgCyclomaticModified R^2: 0.316756, MSE: 0.130860, q: 0.14

Nos testes considerando o limite da taxa de falsas descobertas, controlado através do q, para grupos de uma variável, passaram 37 de 66 hipóteses. Nos grupos de duas variáveis, passaram 939 de 2145 hipóteses. Nos grupos de três variáveis, passaram 19902 de 45760 hipóteses. Contudo, esses resultados ainda requerem analisar o R^2.

Pode-se ver que alguns dos melhores resultados envolveram métricas como CountLineComment (número de linhas com comentários), que não influem no código gerado. Isso pode ter ocorrido por duas razões: o q utilizado ainda é muito alto (em todos resultados, têm-se FDR de, pelo menos, 13 %) ou isso ocorreu pela métrica refletir alguma característica de outra métrica que afeta o código. O número de linhas de comentários provavelmente acompanha o crescimento do número de linhas de código, e o número de ponto e vírgulas provavelmente acompanha o número de *statements*. Por outro lado, outros resultados mais prováveis de influírem no ΔIPC, como as várias métricas de complexidade, apareceram em vários dos melhores candidatos a estarem correlacionados com o ΔIPC.

7 CONCLUSÃO

Não foi possível verificar a existência de correlação entre as métricas de qualidade de software selecionadas e a variação de instruções por ciclo entre as diferentes configurações de x86 analisadas. Atribui-se a isso o pequeno número de *benchmarks* utilizados, que, aliado ao grande número de métricas de qualidade, dificulta verificar a existência de correlação.

Contudo, também não foi possível afirmar que não há relação. Algumas das métricas puderam explicar 30 % da variação de IPC no conjunto de *benchmarks* utilizado. Mesmo ainda não podendo-se concluir que isso vale para softwares em geral, é um indício que pode haver correlação entre as métricas de qualidade e as métricas físicas do software - mesmo que talvez não seja possível explicar completamente as métricas físicas a partir das métricas de qualidade.

Trabalhos futuros podem envolver uma maior variedade de *benchmarks*, análise mais aprofundada das métricas de qualidade, comparações com diferentes arquiteturas ou comparações com diferentes linguagens. Outro aspecto não explorado que pode ter influência nos resultados é o uso de outros compiladores, como Clang ou Intel C++ Compiler.

8 ANEXOS

8.1 Entradas utilizadas

Crypto++ AES

- chave: "9f9f90dbe3e5ee1218c86b8839db1995", SPEC/crypto/fredmans21.txt
- chave: "9f9f90dbe3e5ee1218c86b8839db1995", SPEC/crypto/track3.mp3

Crypto++ DES

- chave: "omega", SPEC/crypto/random96.dat

Crypto++ RSA

- seed: "omega", SPEC/crypto/random96.dat
- seed: "omega", SPEC/crypto/random65536.dat

Crypto++ sign/verify

- SPEC/crypto/random1024.dat
- SPEC/crypto/random65536.dat
- SPEC/crypto/random1048576.dat

cjpeg

- -dct int -quality 90 -outfile /dev/null MediaBench/input_base_4CIF.ppm

djpeg

- -dct int -ppm -outfile /dev/null MediaBench/input_base_4CIF_96bps.jpg

gzip

- compactação e descompactação do resultado SPEC/compress/input/202.tar
- compactação e descompactação do resultado SPEC/compress/input/205.tar
- compactação e descompactação do resultado SPEC/compress/input/208.tar
- compactação e descompactação do resultado SPEC/compress/input/209.tar
- compactação e descompactação do resultado SPEC/compress/input/210.tar

- compactação e descompactação do resultado SPEC/compress/input/211.tar
- compactação e descompactação do resultado SPEC/compress/input/213x.tar
- compactação e descompactação do resultado SPEC/compress/input/228.tar
- compactação e descompactação do resultado SPEC/compress/input/239.tar
- compactação e descompactação do resultado SPEC/compress/input/misc.tar

h263dec

- -o3 MediaBench/input_base_4CIF_96bps.263

h263enc

- -x 4 -a 0 -b 8 -s 15 -G -R 30.00 -r 3508000 -S 3 -Z 30.0 -O 0 -i MediaBench/input_base_4CIF_0to8.yuv -o /dev/null -B output_base_4CIF_96bps_15.263

h264dec

- MediaBench/input_base_4CIF_96bps_decoder.cfg

h264enc

- -d MediaBench/input_base_4CIF_96bps_encoder.cfg

jpg2000dec

- -f MediaBench/input_base_4CIF_96bps.jp2 -F MediaBench/output.ppm -T pnm

jpg2000enc

- -f MediaBench/input_base_4CIF_96bps.ppm -F MediaBench/output.jp2 -T jp2 -O rate=0.010416667

links

- -dump index.html, onde index.html é uma cópia local de http://www.inf.ufrgs.br

mibenchblowfish

- encriptar e decriptar SPEC/crypto/fredmans21.txt com a chave 9f9f90dbe3e5ee1218c86b8839db1995
- encriptar e decriptar SPEC/crypto/track3.mp3 com a chave 9f9f90dbe3e5ee1218c86b8839db1995

mibenchrijndael

- encriptar e decriptar SPEC/crypto/fredmans21.txt com a chave 9f9f90dbe3e5ee1218c86b8839db1995
- encriptar e decriptar SPEC/crypto/track3.mp3 com a chave 9f9f90dbe3e5ee1218c86b8839db1995

mpeg1

- arquivo mp3 de 128kbps com 1m de duração

- arquivo mp3 de 128kbps com 1s de duração
- arquivo mp3 de 128kbps com 100ms de duração

mpeg2decode

- -b MediaBench/input_base_4CIF_96bps.mpg

mpeg2decode_cpp

- -b MediaBench/input_base_4CIF_96bps.mpg

mpeg2encode

- MediaBench/input_base_4CIF_96bps.par

Sablotron

- XML: SPEC/xml.transform/chess-fo/Kasparov-Karpov.xml
  XSD: SPEC/chess-fo/chess.xsl
- XML: SPEC/xml.transform/jenitennison/index.xml
  XSD: SPEC/jenitennison/page.xsl
- XML: SPEC/xml.transform/jenitennison/text.xml
  XSD: SPEC/jenitennison/markup.xsl
- XML: SPEC/xml.transform/nitf/nitf-fishing.xml
  XSD: SPEC/nitf/nitf-stylized.xsl
- XML: SPEC/xml.transform/shared/REC-xml-19980210.xml
  XSD: SPEC/spec-html/xmlspec.xsl
- XML: SPEC/xml.transform/recipes/recipes.xml
  XSD: SPEC/recipes/recipes.xsl
- XML: SPEC/xml.transform/dsd/article.xml
  XSD: SPEC/dsd/article2html.xsl
- XML: SPEC/xml.transform/renderx/chess/Kasparov-Karpov.xml
  XSD: SPEC/renderx/chess/chess.xsl
- XML: SPEC/xml.transform/renderx/examples/balance/balance_sheet.xml
  XSD: SPEC/renderx/examples/balance/balance_sheet.xsl
- XML: SPEC/xml.transform/renderx/examples/meeting/meeting_minutes.xml
  XSD: SPEC/renderx/examples/meeting/meeting_minutes.xsl

sha

- SPEC/crypto.signverify/validity.crypto.signverify.dat

Xerces

- XML: validation_input.xml
  XSD: SPEC/validation_input.xsd
- XML: periodicxsd.xml
  XSD: SPEC/periodic_table.xsd

- XML: much_adoxsd.xml
  XSD: SPEC/play.xsd
- XML: structure.xml
  XSD: SPEC/structure.xsd
- XML: po.xml
  XSD: SPEC/po.xsd
- XML: personal.xml
  XSD: SPEC/personal.xsd

8.2 Scripts em Python e Matlab para Cálculos Estatísticos

```
#!/usr/bin/env python
what=3
s = open('out_bh%d'%(what)).read().split('\n')
#name pval r2 mse
s = [[ y for y in x.strip('␣').split('␣') if y ] for x in s
    if x.strip('␣')]
for i in range(len(s)):
  x = s[i]
  for j in range(len(x)):
    if x[j][0] in '1234567890': break
  x = [':'.join(x[:j])] + x[j:]
  s[i] = x

if what == 1:
  #37 of 66
  crit_p = 0.072557
elif what == 2:
  # 939 of 2145
  crit_p = 0.065637
elif what == 3:
  # 19902 of 45760
  crit_p = 0.060881

m = len(s)


results = []
for x in s:
  x = [x[0]] + [float(x) for x in x[1:]]
  results.append(x)
results.sort(key=lambda x:x[1])

end = False
passed = []
for q in range(1,101):
  q *= 0.01
```

```
  for i in range(len(results)):
    x = results[i]
    i  = i+1
    p = x[1]
    if q >= m*x[1]/i:
      if not end:
        print('q: %f'%(q))
      end = True
      passed.append([x,m*x[1]/i])
  if end: break
if passed:
  passed.sort(key=lambda x:-x[0][2])
for x in passed:
  x, mpi = x
  #x = x[:1] + x[2:] # remove pval
  x = [x[0].replace('_',' &')] + x[1:]
  #x = [x[0]]+[x[2]]
  print('\\item[%s] $R^2$: %f, MSE: %f \\\\'%(x[0].replace('
    &',','),x[2],x[3]))
  #print('%s & %s\\\\'%(x[0], ' & '.join([str(y) for y in x
    [1:]+ ['%.6f'%(mpi)]])))
  #print('%s & %s\\\\'%(x[0], ' & '.join([str(y) for y in x
    [1:]])))
```

```
#!/usr/bin/env python
what = 2
s = open('out%d'%(what)).read()

s = '''0.071365  0.134484  0.165770
   CountDeclInstanceVariableProtected
0.033624  0.181668  0.156733  Essential
0.016576  0.225051  0.148424  CountLineCodeExe
0.034179  0.180650  0.156928  CountOutput
0.034783  0.179560  0.157137  CountLine
0.013754  0.236279  0.146274  CountDeclInstanceVariable
0.061740  0.143610  0.164023  CountDeclInstanceVariablePrivate
0.069098  0.136519  0.165381  AltAvgLineCode
0.044340  0.164404  0.160040  MaxCyclomatic
0.032477  0.183823  0.156321  AltAvgLineBlank
0.034346  0.180346  0.156987  AvgLineBlank
0.040511  0.170054  0.158958  AvgCyclomatic
0.054869  0.151034  0.162601  CountDeclMethodPrivate
0.038936  0.172530  0.158484  CountStmt
0.055221  0.150631  0.162678  CountDeclMethodFriend
0.049898  0.157000  0.161458  SumEssential
0.031296  0.186121  0.155880  CountInput
0.072048  0.133884  0.165885  CountLinePreprocessor
0.022966  0.205206  0.152225  AltCountLineCode
0.040587  0.169937  0.158980  CountLineComment
```

```
0.021374 0.209603 0.151383 CountLineCodeDecl
0.057596 0.147983 0.163185 CountStmtDecl
0.056146 0.149586 0.162878 CountDeclClassVariable
0.034043 0.180897 0.156881 CountLineCode
0.033233 0.182393 0.156594 AltCountLineComment
0.016990 0.223561 0.148710 MaxNesting
0.072159 0.133788 0.165904 CountClassCoupled
0.017064 0.223297 0.148760 CountSemicolon
0.040173 0.170578 0.158857 AvgCyclomaticStrict
0.039235 0.172052 0.158575 CountLineBlank
0.043838 0.165117 0.159903 Knots
0.072557 0.133441 0.165970 AvgLineCode
0.064143 0.141205 0.164483 AvgLine
0.025259 0.199361 0.153345 CountStmtExe
0.072139 0.133805 0.165901 CountDeclFunction
0.031031 0.186649 0.155779 AltCountLineBlank
0.040490 0.170086 0.158952 AvgCyclomaticModified'''
what = 1

s = [x.strip('␣') for x in s.split('\n') if x.strip('␣')]

results = {}
for x in s:
  a = x.split('␣')
  b = a[-1]
  a = a[:-1]
  b = b[b.rfind('/')+1:]
  b = b.replace('.m','')
  b = b.split('_')
  b.sort()
  b = '_'.join(b)
  results[b] = a
sel = 'AvgCyclomatic␣␣␣␣CountClassBase␣␣␣␣CountClassCoupled
   ␣␣␣␣CountClassDerived␣␣␣␣CountInput␣␣␣␣CountOutput␣␣␣␣
   MaxCyclomatic␣␣␣␣MaxInheritanceTree␣␣␣␣MaxNesting␣␣␣␣
   PercentLackOfCohesion␣␣␣␣SumCyclomatic'
sel = [ x for x in sel.split('␣') if x ]

keys = []
for x in sel:
  if what == 1:
    a = x
    if a not in keys:
      keys.append(a)
  elif what == 2:
    for y in sel:
      if y != x:
        a = [x, y]
```

```
        a.sort()
        a = '_'.join(a)
        if a not in keys:
          keys.append(a)
  elif what == 3:
    for y in sel:
      if y == x: continue
      for z in sel:
        if z == x or z == y: continue
        a = [x, y, z]
        a.sort()
        a = '_'.join(a)
        if a not in keys:
          keys.append(a)

l = []
for x in keys:
  if x in results:
    l.append((results[x], x))
l = [(results[x],x) for x in results]
l.sort(key=lambda x:-float(x[0][1]))
print(len(l))
for x in l[:min(len(l),15)]:
  a,b = x
  b = b.split('_')
  print('%s & %s \\\\'%(' & '.join(b),' & '.join(a)))
```

```
#!/usr/bin/env python
import os
import sys
import csv

gOutputDir = 'combinations'

def combine(l, n):
  if n == 0:
    return []
  if n == 1:
    return [ [x] for x in l ]
  results = []
  for i in range(len(l)):
    if len(l[i+1:]) < n - 1:
      break
    results += [ [ l[i] ] + x for x in combine(l[i+1:], n
      -1) ]
  return results

def load_metrics():
  f = open('metrics.csv', 'rt')
```

```
  lines = f.readlines()
  f.close()
  l = []
  rows = [ x for x in csv.reader(lines) ]

  names = [x for x in rows[0]]
  for i in range(len(rows[0])):
    if rows[0][i] == '-':
      divider = i
      for i in range(divider+1, len(rows[0])):
        names[i] = names[i]+'X'
      names.pop(divider)
      for r in rows:
        r.pop(divider)
      break

  metrics = {}
  for name in names:
    metrics[name] = []

  for row in rows:
    for i in range(len(names)):
      metrics[names[i]].append(row[i])
  metrics.pop('name')

  # filter out metrics where all values are zero
  for name in [x for x in metrics]:
    m = metrics[name][1:]
    if not [x for x in m if float(x)!=0]:
      metrics.pop(name)

  return metrics

def main():
  if not os.path.isdir(gOutputDir):
    os.mkdir(gOutputDir)
  fnames = []
  metrics = load_metrics()
  metricsNames = [x for x in metrics]
  nn = 1
  for nn in range(nn,nn+1):
    dstDir = os.path.join(gOutputDir, str(nn))
    if not os.path.isdir(dstDir):
      os.mkdir(dstDir)
    for comb in combine(metricsNames, nn):
      fname = os.path.join(dstDir, '_'.join(comb) + '.m')
      fnames.append(fname)
      if not os.path.isfile(fname):
```

```
            buf = ''
            for i in range(1, len(metrics[comb[0]])):
              buf += '1,\t'
              for name in comb:
                buf += '%s,\t'%(metrics[name][i])
              buf = buf[:-2] + ';\n'
            f = open(fname, 'wt')
            f.write(buf)
            f.close()
    buf = '''y = [   1.7177
    1.6039
    1.6121
    0.8125
    1.3310
    1.4588
    0.5112
    1.1958
    1.8934
    1.0657
    1.1510
    0.8013
    0.7964
    1.1054
    1.7930
    2.1217
    1.2562
    1.1769
    1.3253
    2.1115
    0.8382
    0.8262
    0.9035
    0.7579
    0.9051];
n = 25;
k = %d;

names = {
%s
};

all_pval = 1:size(names,1);
all_r2 = 1:size(names,1);
all_sse = 1:size(names,1);
for i=1:size(names),
  name = names(i,:);
  name = name{:};
  x = load(name);
```

```
%%  b = regress(y, x);
%%  ssreg = sum( ( (x*b) - mean(y) ) .^ 2);
%%  sstot = sum(y.^2) - (sum(y)^2) / max(size(y));

   [b,bint,r,rint,stats] = regress(y, x);
   ssyy = sum( (y - mean(y)) .^ 2 );
   sse = sum( (y - (x * b)) .^ 2 );
   r2 = 1 - sse / ssyy;
   f = (r2 / k) / ( (1 - r2) / (n - (k + 1)) );
%%  fprintf(\'%%f %%f %%s\\n\', f, r2, name);
%%  fprintf(\'%%f %%f %%s\\n\', r2, sse/n, name);
   pval = stats(3);
   all_pval(i) = pval;
   all_r2(i) = r2;
   all_sse(i) = sse;
%%  fprintf(\'%%f %%f %%f %%s\\n\', pval, r2, sse/n, name);



%%  b = regress(y, x);
%%  b1 = b(2:end);
%%  sxx = sum( (x(:,2) - mean(x(:,2))) .^ 2);
%%  sse = sum( (y - x*b) .^ 2);
%%  v = sse / (n - (k+1));
%%  t0 = b1 / ( (v / sxx) ^ 0.5);
%%  fprintf(\'%%f %%s\\n\', t0, name);

%% check if we can use a parametric test
%%  s = (sum( (x(:,2) - mean(x(:,2))) .^ 2 ) / (n-1))^0.5;
%%  simmetry = skewness(x(:,2)) / s;
%%  mesocurtica = kurtosis(x(:,2)) / s;
%%  fprintf(\'%%f %%s\\n\', mesocurtica, name);

%% spearman's rho
%%  [rho, pval] = corr(x(:,2), y, 'type', 'Spearman');
%%  fprintf(\'%%f %%s\\n\', rho, name);
end;

for q=0.01*[1:100],
   [h crit_p adj_p] = fdr_bh(all_pval, q, 'pdep', 'yes');
   if crit_p ~= 0
     break;
   end;
end;
fprintf(\'crit_p %%f\\n\', crit_p);
fprintf(\'pval r2 mse\\n\');
for i=1:max(size(all_pval))
```

```
    fprintf(\'\\n\');
    s='';
    for j=1:1
        r=names(i,:);
        r=r{:};
        r=r(1,16:end-2);
        s=[s r ' '];
    end;
    fprintf(\'%%s %%f %%f %%f\\n\',s,all_pval(i),all_r2(i),
        all_sse(i)/n);
end;
exit;
%%for i=1:max(size(all_pval))
%%   if all_pval(i) <= crit_p
%%      s = '';
%%      for j=1:k
%%         r = names(i);
%%         r = r{:};
%%         r = r(1,16:end-2);
%%         s = [s r ' '];
%%      end;
%%%%pval, r2, MSE
%%      fprintf(\'%%f %%f %%f %%s\\n\',all_pval(i), all_r2(i)
    , all_sse(i)/n, s);
%%   end;
%%end;

%%exit;
'''%(nn, ',\n'.join(['\''+x+'\'' for x in fnames]))
    f = open('go.m', 'wt')
    f.write(buf)
    f.close()

main()

prNames = {'cjpeg'
    'cryptopp_aes'
    'cryptopp_des'
    'cryptopp_rsa'
    'cryptopp_signverify'
    'djpeg'
    'gzip'
    'h263dec'
    'h263enc'
    'h264dec'
    'h264enc'
    'jpg2000dec'
    'jpg2000enc'
    'links'
```

```
   'mibenchblowfish'
   'mibenchrijndael'
   'mpeg1'
   'mpeg2decode'
   'mpeg2decode_cpp'
   'mpeg2encode'
   'sablot'
   'sha'
   'xerces'
   'xercessax'
   'xercesdom'
};

y = [ 1.7177
   1.6039
   1.6121
   0.8125
   1.3310
   1.4588
   0.5112
   1.1958
   1.8934
   1.0657
   1.1510
   0.8013
   0.7964
   1.1054
   1.7930
   2.1217
   1.2562
   1.1769
   1.3253
   2.1115
   0.8382
   0.8262
   0.9035
   0.7579
   0.9051
];
n = 25;
k = 1;

names = {
'combinations/1/Cyclomatic.m',
'combinations/1/MaxEssentialKnotsX.m',
'combinations/1/MaxEssentialKnots.m',
'combinations/1/CountDeclMethodProtectedX.m',
'combinations/1/CountStmtDeclX.m',
```

```
'combinations/1/CountClassCoupledX.m',
'combinations/1/AltCountLineCode.m',
'combinations/1/KnotsX.m',
'combinations/1/CountDeclMethodPublic.m',
'combinations/1/CountLineX.m',
'combinations/1/AltAvgLineCode.m',
'combinations/1/CountDeclInstanceVariablePrivate.m',
'combinations/1/MinEssentialKnots.m',
'combinations/1/CountDeclMethodAll.m',
'combinations/1/MaxCyclomaticStrictX.m',
'combinations/1/MaxCyclomatic.m',
'combinations/1/CountStmt.m',
'combinations/1/AvgLineComment.m',
'combinations/1/CountDeclInstanceMethod.m',
'combinations/1/CountDeclMethodFriend.m',
'combinations/1/MaxNesting.m',
'combinations/1/SumEssentialX.m',
'combinations/1/SumCyclomaticX.m',
'combinations/1/AvgLineCommentX.m',
'combinations/1/MaxCyclomaticModified.m',
'combinations/1/CountLineCodeExeX.m',
'combinations/1/SumCyclomatic.m',
'combinations/1/SumEssential.m',
'combinations/1/CountStmtEmpty.m',
'combinations/1/AvgLineCode.m',
'combinations/1/AltCountLineCommentX.m',
'combinations/1/CountLinePreprocessor.m',
'combinations/1/MaxInheritanceTree.m',
'combinations/1/CountLineCode.m',
'combinations/1/AvgEssentialX.m',
'combinations/1/CountDeclMethod.m',
'combinations/1/AltCountLineBlank.m',
'combinations/1/CountPath.m',
'combinations/1/CountLineComment.m',
'combinations/1/AvgCyclomaticStrictX.m',
'combinations/1/RatioCommentToCode.m',
'combinations/1/EssentialX.m',
'combinations/1/MinEssentialKnotsX.m',
'combinations/1/CountLineInactive.m',
'combinations/1/CountStmtEmptyX.m',
'combinations/1/AvgLineBlankX.m',
'combinations/1/MaxCyclomaticModifiedX.m',
'combinations/1/AvgLine.m',
'combinations/1/CountDeclInstanceVariableX.m',
'combinations/1/SumCyclomaticStrict.m',
'combinations/1/CyclomaticModifiedX.m',
'combinations/1/CountDeclClass.m',
'combinations/1/CountDeclClassVariableX.m',
```

```
'combinations/1/AvgLineCodeX.m',
'combinations/1/CountStmtExe.m',
'combinations/1/CountInputX.m',
'combinations/1/CountDeclMethodProtected.m',
'combinations/1/CyclomaticX.m',
'combinations/1/CountStmtExeX.m',
'combinations/1/CountLineCodeX.m',
'combinations/1/CountLineCommentX.m',
'combinations/1/SumCyclomaticModified.m',
'combinations/1/AvgLineBlank.m',
'combinations/1/CountDeclMethodPublicX.m',
'combinations/1/CountDeclInstanceVariableProtected.m',
'combinations/1/CountDeclMethodPrivateX.m',
'combinations/1/MaxCyclomaticX.m',
'combinations/1/CountDeclInstanceVariablePrivateX.m',
'combinations/1/CountDeclClassMethod.m',
'combinations/1/MaxInheritanceTreeX.m',
'combinations/1/CountDeclMethodX.m',
'combinations/1/CountClassBaseX.m',
'combinations/1/CountLinePreprocessorX.m',
'combinations/1/AltAvgLineBlank.m',
'combinations/1/SumCyclomaticModifiedX.m',
'combinations/1/CountDeclMethodFriendX.m',
'combinations/1/CountLineCodeExe.m',
'combinations/1/CountStmtDecl.m',
'combinations/1/CountDeclFunction.m',
'combinations/1/CountSemicolon.m',
'combinations/1/AvgCyclomaticX.m',
'combinations/1/CountLineBlankX.m',
'combinations/1/CountSemicolonX.m',
'combinations/1/CountLineCodeDeclX.m',
'combinations/1/AvgEssential.m',
'combinations/1/CountDeclMethodConst.m',
'combinations/1/RatioCommentToCodeX.m',
'combinations/1/AvgCyclomaticModifiedX.m',
'combinations/1/Essential.m',
'combinations/1/MaxCyclomaticStrict.m',
'combinations/1/SumCyclomaticStrictX.m',
'combinations/1/MaxNestingX.m',
'combinations/1/CyclomaticModified.m',
'combinations/1/CountDeclInstanceVariableProtectedX.m',
'combinations/1/AltAvgLineComment.m',
'combinations/1/CountDeclClassMethodX.m',
'combinations/1/CountDeclClassVariable.m',
'combinations/1/AvgLineX.m',
'combinations/1/CountDeclInstanceVariablePublic.m',
'combinations/1/CountLine.m',
'combinations/1/CountClassDerivedX.m',
```

```
'combinations/1/PercentLackOfCohesionX.m',
'combinations/1/CountDeclFile.m',
'combinations/1/CountDeclInstanceVariable.m',
'combinations/1/CountLineBlank.m',
'combinations/1/AltCountLineCodeX.m',
'combinations/1/AvgCyclomatic.m',
'combinations/1/CountLineInactiveX.m',
'combinations/1/CyclomaticStrictX.m',
'combinations/1/CyclomaticStrict.m',
'combinations/1/CountDeclMethodConstX.m',
'combinations/1/CountDeclFileHeader.m',
'combinations/1/AltCountLineComment.m',
'combinations/1/CountLineCodeDecl.m',
'combinations/1/CountDeclMethodAllX.m',
'combinations/1/CountDeclFileCode.m',
'combinations/1/CountOutputX.m',
'combinations/1/AvgCyclomaticModified.m',
'combinations/1/Knots.m',
'combinations/1/CountDeclInstanceMethodX.m',
'combinations/1/CountDeclInstanceVariablePublicX.m',
'combinations/1/CountStmtX.m',
'combinations/1/CountDeclMethodPrivate.m',
'combinations/1/AvgCyclomaticStrict.m',
'combinations/1/CountPathX.m'
};

values = zeros(max(size(y)), max(size(names)));
for i=1:size(names),
  name = names(i);
  name = name{:};
  if exist(name)
    x = load(name);
    values(:,i) = x(:,2);
  end;
end;
numVal = size(values,2);

function [y] = has_null_col(m)
n = size(m);
if n(1) == 1
  y = sum(m==0);
else
  y = sum(sum(m)==0);
end;

function [stats] = get_stats()
load_data;
config;
fname = sprintf('stats_data_%d_%d.mat', numPredictors,
```

```
    numVal);
load(fname, 'results');
stats = results;

function [names stats values] = get_selected_data(names,
    stats, values, numPredictors)
goodnames = {
%'AvgCyclomatic',
%'CountClassBase',
%'CountClassCoupled',
%'CountClassDerived',
%'CountInput',
%'CountOutput',
%'MaxCyclomatic',
%'MaxInheritanceTree',
%'MaxNesting',
%'PercentLackOfCohesion',
%'SumCyclomatic',

%'combinations/1/AvgCyclomatic.m',
'combinations/1/CountClassBaseX.m',
'combinations/1/CountClassCoupledX.m',
'combinations/1/CountClassDerivedX.m',
'combinations/1/CountInputX.m',
'combinations/1/CountOutputX.m',
'combinations/1/MaxCyclomatic.m',
'combinations/1/MaxInheritanceTree.m',
'combinations/1/MaxNesting.m',
'combinations/1/PercentLackOfCohesionX.m',
%'combinations/1/SumCyclomatic.m',
};

goodnames = {
'combinations/1/AltAvgLineCode.m',
'combinations/1/AltCountLineCode.m',
'combinations/1/AvgCyclomatic.m',
'combinations/1/AvgCyclomaticModified.m',
'combinations/1/AvgCyclomaticStrict.m',
'combinations/1/AvgEssential.m',
'combinations/1/AvgLine.m',
'combinations/1/AvgLineCode.m',
'combinations/1/CountClassBase.m',
'combinations/1/CountClassCoupled.m',
'combinations/1/CountClassDerived.m',
'combinations/1/CountDeclClass.m',
'combinations/1/CountDeclClassMethod.m',
'combinations/1/CountDeclClassVariable.m',
'combinations/1/CountDeclFunction.m',
'combinations/1/CountDeclInstanceMethod.m',
```

```
'combinations/1/CountDeclInstanceVariable.m',
'combinations/1/CountDeclInstanceVariablePrivate.m',
'combinations/1/CountDeclInstanceVariableProtected.m',
'combinations/1/CountDeclInstanceVariablePublic.m',
'combinations/1/CountDeclMethod.m',
'combinations/1/CountDeclMethodAll.m',
'combinations/1/CountDeclMethodConst.m',
'combinations/1/CountDeclMethodFriend.m',
'combinations/1/CountDeclMethodPrivate.m',
'combinations/1/CountDeclMethodProtected.m',
'combinations/1/CountDeclMethodPublic.m',
'combinations/1/CountInput.m',
'combinations/1/CountLineCode.m',
'combinations/1/CountLineCodeDecl.m',
'combinations/1/CountLineCodeExe.m',
'combinations/1/CountLinePreprocessor.m',
'combinations/1/CountOutput.m',
'combinations/1/CountPath.m',
'combinations/1/CountStmt.m',
'combinations/1/CountStmtDecl.m',
'combinations/1/CountStmtEmpty.m',
'combinations/1/CountStmtExe.m',
'combinations/1/Cyclomatic.m',
'combinations/1/CyclomaticModified.m',
'combinations/1/CyclomaticStrict.m',
'combinations/1/Essential.m',
'combinations/1/Knots.m',
'combinations/1/MaxCyclomatic.m',
'combinations/1/MaxCyclomaticModified.m',
'combinations/1/MaxCyclomaticStrict.m',
'combinations/1/MaxEssentialKnots.m',
'combinations/1/MaxInheritanceTree.m',
'combinations/1/MaxNesting.m',
'combinations/1/PercentLackOfCohesion.m',
'combinations/1/SumCyclomatic.m',
'combinations/1/SumCyclomaticModified.m',
'combinations/1/SumCyclomaticStrict.m',
'combinations/1/SumEssential.m',
};

badindexes = [];
i = size(names,1);
while i > 0
 s = names(i);
 s = s{:};
 ok = 0;
 for j=1:size(goodnames)
  r = goodnames(j);
```

```
   r = r{:};
   if findstr(s, r)
    ok = 1;
    break;
   end;
  end;
  if ~ok
%   names = [ names(1:i-1) ; names(i+1:end) ];
%   values = [ values(:, 1:i-1) values(:, i+1:end) ];
%   badindexes = [badindexes i];
  end;
  i = i - 1;
end;
badindexes = sort(badindexes);

for i=1:max(size(badindexes))
 i = badindexes(i);
 for j=1:numPredictors
  stats = stats(stats(:,j) ~= i, :);
 end;
end;

for i=1:max(size(badindexes))
 badindexes(i+1:end) = badindexes(i+1:end) - 1;
 for j=1:size(stats,1)
  for k=1:numPredictors
   if stats(j,k) >= badindexes(i)
    stats(j,k) = stats(j,k) - 1;
   end;
  end;
 end;
end;

config;
load_data;
numVal = size(values, 2);
x = zeros(size(values,12), numPredictors);
chosen = 1:numPredictors;
if numPredictors > 1
 chosen(end) = chosen(end-1);
else
 chosen(end) = 0;
end;

results = zeros(nchoosek(numVal, numPredictors), size(
   chosen,2) + 2);
resultNum = 1;

totalResults = size(results, 1);
```

```
allChosen = choose(totalResults, numPredictors, numVal);

while resultNum <= size(results, 1),
 for i=numPredictors+1-(1:numPredictors),
  chosen(i) = chosen(i) + 1;
  if(chosen(i) <= numVal & chosen(i) + numPredictors - i <=
      numVal)
   for i=(i+1):numPredictors
    chosen(i) = chosen(i-1)+1;
   end;
   break;
  end;
 end;
 if sum(chosen ~= allChosen(resultNum,:)),
  fprintf('bad\n');
 end;
 x = values(:,chosen);
 stat = regstats(y,x);
 r2 = stat.rsquare;
 p = stat.tstat.pval;
 results(resultNum,:) = [ chosen r2 max(p) ];
 resultNum  = resultNum + 1;
 chosen
end;
fname = sprintf('stats_data_%d_%d.mat',numPredictors,
   numVal);
save(fname, 'results');

y = [ 1.7177
   1.6039
   1.6121
   0.8125
   1.3310
   1.4588
   0.5112
   1.1958
   1.8934
   1.0657
   1.1510
   0.8013
   0.7964
   1.1054
   1.7930
   2.1217
   1.2562
   1.1769
   1.3253
   2.1115
   0.8382
```

```
   0.8262
   0.9035
   0.7579
   0.9051];
n = 25;
k = 1;

names = {
'combinations/1/CountDeclInstanceVariablePrivate.m',
'combinations/1/AltAvgLineComment.m',
'combinations/1/CountLineComment.m',
'combinations/1/CyclomaticStrict.m',
'combinations/1/MaxEssentialKnots.m',
'combinations/1/Essential.m',
'combinations/1/AltCountLineComment.m',
'combinations/1/CountDeclMethodFriend.m',
'combinations/1/AltAvgLineBlank.m',
'combinations/1/CountPath.m',
'combinations/1/CountLineBlank.m',
'combinations/1/MaxCyclomatic.m',
'combinations/1/AvgLineCode.m',
'combinations/1/AltAvgLineCode.m',
'combinations/1/CountLineCode.m',
'combinations/1/CountDeclFunction.m',
'combinations/1/Cyclomatic.m',
'combinations/1/CountLineInactive.m',
'combinations/1/AvgLineComment.m',
'combinations/1/Knots.m',
'combinations/1/AltCountLineBlank.m',
'combinations/1/CountDeclClass.m',
'combinations/1/MaxCyclomaticStrict.m',
'combinations/1/AvgCyclomaticModified.m',
'combinations/1/CountDeclInstanceMethod.m',
'combinations/1/CountStmtExe.m',
'combinations/1/CountStmtEmpty.m',
'combinations/1/CountClassCoupled.m',
'combinations/1/CountDeclClassVariable.m',
'combinations/1/SumCyclomaticStrict.m',
'combinations/1/CountSemicolon.m',
'combinations/1/SumCyclomaticModified.m',
'combinations/1/CountDeclClassMethod.m',
'combinations/1/AvgLine.m',
'combinations/1/CountLineCodeExe.m',
'combinations/1/CountDeclInstanceVariablePublic.m',
'combinations/1/SumCyclomatic.m',
'combinations/1/CountDeclMethodConst.m',
'combinations/1/AvgCyclomaticStrict.m',
'combinations/1/AvgCyclomatic.m',
```

```
'combinations/1/PercentLackOfCohesion.m',
'combinations/1/CountDeclInstanceVariable.m',
'combinations/1/MaxCyclomaticModified.m',
'combinations/1/CountClassDerived.m',
'combinations/1/CountLineCodeDecl.m',
'combinations/1/CountLinePreprocessor.m',
'combinations/1/CyclomaticModified.m',
'combinations/1/SumEssential.m',
'combinations/1/CountDeclMethodProtected.m',
'combinations/1/CountDeclMethodAll.m',
'combinations/1/MaxNesting.m',
'combinations/1/AltCountLineCode.m',
'combinations/1/RatioCommentToCode.m',
'combinations/1/CountDeclInstanceVariableProtected.m',
'combinations/1/CountStmt.m',
'combinations/1/CountDeclMethodPrivate.m',
'combinations/1/CountDeclMethod.m',
'combinations/1/CountDeclMethodPublic.m',
'combinations/1/AvgEssential.m',
'combinations/1/CountStmtDecl.m',
'combinations/1/CountInput.m',
'combinations/1/CountLine.m',
'combinations/1/CountOutput.m',
'combinations/1/CountClassBase.m',
'combinations/1/MaxInheritanceTree.m',
'combinations/1/AvgLineBlank.m'
};

for i=1:size(names),
  name = names(i,:);
  name = name{:};
  x = load(name);

% b = regress(y, x);
% ssreg = sum( ( (x*b) - mean(y) ) .^ 2);
% sstot = sum(y.^2) - (sum(y)^2) / max(size(y));

  b = regress(y, x);
  ssyy = sum( (y - mean(y)) .^ 2 );
  sse = sum( (y - (x * b)) .^ 2 );
  r2 = 1 - sse / ssyy;
  f = (r2 / k) / ( (1 - r2) / (n - (k + 1)) );
% fprintf('%f %f %s\n', f, r2, name);
  fprintf('%f␣%s\n', r2, name);


% b = regress(y, x);
% b1 = b(2:end);
```

```
% sxx = sum( (x(:,2) - mean(x(:,2))) .^ 2);
% sse = sum( (y - x*b) .^ 2);
% v = sse / (n - (k+1));
% t0 = b1 / ( (v / sxx) ^ 0.5);
% fprintf('%f %s\n', t0, name);

% check if we can use a parametric test
% s = (sum( (x(:,2) - mean(x(:,2))) .^ 2 ) / (n-1))^0.5;
% simmetry = skewness(x(:,2)) / s;
% mesocurtica = kurtosis(x(:,2)) / s;
% fprintf('%f %s\n', mesocurtica, name);

% spearman's rho
% [rho, pval] = corr(x(:,2), y, 'type', 'Spearman');
% fprintf('%f %s\n', rho, name);
end;
```

8.3 Scripts para Obtenção de Métricas

```
#!/usr/bin/env pypy
'''
KNOWN BUGS
  Valgrind gives us the names of some global objects (
     global
  constructors keyed to ...), instead of their classes.
     Some/most/all of these
  are, in fact, variables and as such shouldn't even have
     constructors.


objdump -x -t -D -l test_aes_des
addr2line
~/src/valgrind/valgrind/funtrace
~/metricas/metricas/profiling/valgrind/src/cryptopp

for cryptopp_aes
  more than one class: all ok
  no classes: all ok
  more than a single match: all ok
  no matches:
    global constructors keyed to CryptoPP::g_actualMac : /
       home/luis/metricas/metricas/profiling/valgrind/src/
       cryptopp/fipstest.cpp : 582 : 0
    global constructors keyed to CryptoPP::DEFAULT_CHANNEL
       : /home/luis/metricas/metricas/profiling/valgrind/
       src/cryptopp/cryptlib.cpp : 826 : 0
    global constructors keyed to CryptoPP::
       PKCS_DigestDecoration<CryptoPP::SHA1>::decoration :
       /home/luis/metricas/metricas/profiling/valgrind/src/
```

```
        cryptopp/dll.cpp : 38 : 0
      global constructors keyed to CLOCK_TICKS_PER_SECOND : /
        home/luis/metricas/metricas/profiling/valgrind/src/
        cryptopp/bench.cpp : 24 : 0
      global constructors keyed to AdhocTest : /home/luis/
        metricas/metricas/profiling/valgrind/src/cryptopp/
        test_aes_des.cpp : 96 : 0
'''
import interactive
import os
import re
import sys
from sys import exit

import pdb

class Id:
  def __init__(self):
    self.id = 1
  def new_id(self):
    self.id += 1
    return self.id - 1
new_id = Id().new_id

import inspect
def lineno():
    """Returns the current line number in our program."""
    return inspect.currentframe().f_back.f_lineno

def filename_fix(s):
  '''
  while s[:2] == './':
    s = s[2:]
  s = s.replace('/./', '/')
  '''
  s = os.path.abspath(s)
  s = s.replace('/home/luis/metricas/metricas/profiling/
     valgrind', '/mnt/lfgmillani/metricas')
  return s

class Symbol:
  def __init__(self, val, typ, name, fname, line):
    self.id = new_id()
    self.val = val
    self.typ = typ
    self.name = name
    self.fname = filename_fix(fname) if fname else fname
    self.line = line
```

```
  def __repr__(self):
    return '%s␣%s␣%s␣%s␣%s␣%s'%(self.id, self.val, self.typ
       , self.name, self.fname, self.line)


def remove_pairs(s, l, r):
  while 1:
    li, ri = None, None
    for i in range(len(s)):
      c = s[i]
      if c == l:
        li = i
      elif c == r:
        ri = i
        if li:
          s = s[:li] + s[ri+1:]
          break
    else:
      break
  return s

def get_very_short_fun_name(s):
  d = {}
  for x in re.findall('operator[^(]+', s):
    d[x] = '__%x__'%(hash(x))
  for x in ('>','>>','->','<','<<'):
    d['operator'+x] = '__%x__'%(hash(x))
  for x in d:
    s = s.replace(x,d[x])
  s = re.sub('^non-virtual␣thunk␣to␣', '', s)
  s = re.sub('^global␣constructors␣keyed␣to␣', '', s)
  s = re.sub('␣const$', '', s)
  s = remove_pairs(s, '(',')')
  s = remove_pairs(s, '<','>')
  if '␣' in s:
    s = s[s.rfind('␣')+1:]
  for x in d:
    s = s.replace(d[x],x)
  return s

def get_nm_data():
  f = open('nmdata', 'rt')
  s = f.read().split('\n')
  f.close()
  d = {}
  for l in s:
    m = re.match('([\\da-f]+)\\s([ABbCDdGgiNpRrSsTtUuVvWw])
       \\s([^\t]+)(?:\\t([^:]*)[:\\s](\\d+))?$', l)
```

```
    if m:
      val, typ, name, fname, line = m.groups()
      val = int(val, 16)
      d[val] = Symbol(val, typ, name, fname, line)
  return d

def nm_to_short_names(nm):
  d = {}
  for x in nm:
    s = get_very_short_fun_name(nm[x].name)
    if s not in d:
      d[s] = []
    d[s].append(nm[x])
  return d

gNm = get_nm_data()
gNmByShortNames = nm_to_short_names(gNm)

class FileEq():
  def __init__(self):
    self.hashes = {}
  def eq(self, a,b):
    for x in (a, b):
      if x not in self.hashes:
        f = open(x)
        data = f.read()
        f.close()
        self.hashes[x] = hash(data)
    return self.hashes[a] == self.hashes[b]

file_is_eq = FileEq().eq

class Function:
  def __init__(self, name, line, dirname, filename = None,
     addr = None):
    self.id = new_id()
    if filename == '???':
      filename = None
    if dirname == '???':
      dirname = None
    self.name = name
    if re.match('.*\\(\\)\\sconst$', self.name):
      self.name = self.name[:self.name.rfind('()␣const')]
    self.filename = filename_fix(os.path.join(dirname,
       filename)) if filename else (filename_fix(dirname)
       if dirname else None)
    assert( type(self.filename) in (str,type(None)) )
    self.addr = addr
```

```
    self.metrics = {}
    self.mark=0
    self.line = line
#   self.line = get_real_line_number_from_und(self)
    if self.addr:
      self.addr = int(self.addr, 16)
      self.get_data_from_nm()
  def get_data_from_nm(self):
  #nm -p -a -C -l test_aes_des
    name = self.name
#   if re.match('.*\\(\\)\\sconst$', name):
#     name = name[:name.rfind('() const')]
    if self.filename and '/build/' == self.filename[:7]:
      print('ignoring lib function: %s'%(self.name))
      return
    '''   WARNING ! WARNING ! WARNING ! WARNING !
          REMEMBER TO REMOVE "HAXHAXHAX" IN THE TWO STRINGS
              BELOW
          WARNING ! WARNING ! WARNING ! WARNING !
    '''
#   if re.search('^(?:(?:global constructors keyed to )|(?:
  HAXHAXHAXnon-virtual thunk to ))', name):
#     noGlobalName = re.sub('^(?:(?:global constructors
  keyed to )|(?:HAXHAXHAXHnon-virtual thunk to ))', '',
  name)
    if re.search('^(?:global constructors keyed to )', name
      ):
      noGlobalName = re.sub('^(?:global constructors keyed
        to )', '', name)
      shortname = get_very_short_fun_name(noGlobalName)
      try:
        l = gNmByShortNames[shortname]
      except KeyError:
        print('failed to find proper line for\n%s'%(self.
          name))
        return
      l = [ x for x in l if x.line ]
      print('!!!',self.filename)
      if self.filename and '/build/' != self.filename[:7]:
        l = [x for x in l if x.fname == self.filename ]
      if [x for x in l if x.name == noGlobalName]:
        l = [x for x in l if x.name == noGlobalName]
      if len(l) == len([x for x in l if x.line == l[0].line
        ]):
        l = l[:1]
      if len(l) > 1:
        print('%d: more than single nm match for\n%s'%(
          lineno(), self.name))
```

```
            print(['%x'%(x.val) for x in l])
            print(['%s'%(x.name) for x in l])
            exit(1)
          elif len(l) == 1:
            self.line = l[0].line
            self.line = get_real_line_number_from_und(self)
        else:
          if self.addr not in gNm:
            print('failed to find symbol %x / %s'%(self.addr,
                self))
            exit(1)
            self.line = -666
            return
          x = gNm[self.addr]
          if self.filename != x.fname and file_is_eq(self.
              filename, x.fname):
            self.filename = x.fname
          if self.filename != x.fname:
            if '~' in self.name: # or 'global constructors
                keyed to ' in self.name:
              return
            print('%d: filename mismatch between funtrace and
                nm for function\n%s\nADDR: %x\nfunfile: %s @ %s\
                nnmfile: %s @ %s'%(lineno(), self.name, self.
                addr,self.filename, self.line, x.fname, x.line))
            exit(1)
          self.line = x.line
          self.line = get_real_line_number_from_und(self)
      def __repr__(self):
        return '%s : %s : %s : %s : %s'%(self.id, self.name,
            self.filename, self.line, self.mark)
      def __cmp__(self, other):
        n = [cmp(x, y) for x,y in [
          (self.addr, other.addr),
          (self.name, other.name),
          (self.filename, other.filename),
          (self.line, other.line),
          (self.metrics, other.metrics),
        ] if cmp(x, y) ]
        if n:
          return n[0]
        return 0

    def parse_calls(fname):
      f = open(fname)
      s = f.read().split('\n')
      f.close()
      results = []
```

```
  for l in s:
    if not l or l[0] == '=': continue
    print(l)
    name, dirname, filename, line, addr = l.split('\t')
    results.append( Function(name, line, dirname, filename,
        addr) )

  d = {}
  for r in results:
#   d['$'.join([r.name, r.filename, r.line])] = r
    if r.addr in d:
      if r != d[r.addr]:
        print('%d: found two lines with same addr but
           differing data:\n%s\n%s'%(
          lineno(),
          d[r.addr],
          r))
        exit(1)
    d[r.addr] = r
  results = [ d[x] for x in d ]
  return results

def get_real_line_number_from_und(u):
  if not (u.line and u.line!='-1' and u.filename): return u
     .line
  if '/build/' == u.filename[:7]: return u.line
  if '__static_initialization_and_destruction_0' in u.name:
      return u.line
  short = get_very_short_fun_name(u.name)
  if '::' in short:
    short = short.split('::')[-1]
  print(short)
  print(u)
  f = open(u.filename)
  lines = f.read().split('\n')
  f.close()
  n = int(u.line)-1

  candidates = [ x for x in range(len(lines)) if short in
     lines[x] ]
  print(n,candidates)
  if n in candidates:
    pass
  elif not candidates and '~' == short[0]:
    pass
  elif not candidates:
    pass
```

```
# elif '{' in lines[n]:
  elif lines[n].rfind('{') > lines[n].rfind('}'):
    m = min([x for x in range(len(lines)) if '{' in lines[x
       ] and x >= n])
#   print(m)
    n = max([x for x in candidates if x <= m])
  else:
    diff = [abs(n-x) for x in candidates]
    m = min(diff)
    for x in range(len(diff)):
      if diff[x] == m:
        n = candidates[x]

  return str(n+1)


  candidates = [ x for x in range(len(lines)) if short in
     lines[x] ]
  if not candidates and '~' == short[0]:
    return u.line
  print(n, lines[n], candidates)
  if len(candidates) == 1:
    n = candidates[0]
  elif '{' in lines[n]:
    n = max([x for x in candidates if x <= n])
  elif not lines[n]:
    print(lineno(),'WTF')
    exit(1)
  else:
    n = min([x for x in candidates if x >= n])
  return str(n+1)

  n = int(u.line)-1
  f = open(u.filename)
  lines = f.read().split('\n')
  f.close()
  if 45521 == u.id:
    print('ZXC',n,lines[n])
    exit(1)
  if lines[n] and lines[n][-1] == '{':
    if 45521 == u.id:
      print('XXXXXXXX')
      exit(1)
    s = '\n'.join(lines[:n+1])
    if 45521 == u.id:
      print(s[-30:])
      exit(1)
    if re.match('.*[)]\\s*[{]$',s,re.S|re.M):
```

```
        par = 0
        while s[-1] != ')':
          if '\n' == s[-1]:
            n -= 1
          s = s[:-1]
        while s[-1] != '(':
          if '\n' == s[-1]:
            n -= 1
          s = s[:-1]
#     print(lineno(), '#',lines[n],n,u.filename)
#     exit(1)
    if 45521 == u.id:
      print('Y'*8,n)
      exit(1)
    return str(n + 1)

    n = int(u.line) - 1
    name = u.name
    if '::' in u.name:
      name = u.name.split('::')[-1]
    f = open(u.filename, 'rt')
    lines = f.read().split('\n')
    f.close()
    for i in range(10):
      for i in (-i,i):
        if n+i >= 0 and n+i < len(lines) and name in lines[n+
            i]:
          while n+i < len(lines) and '{' not in lines[n+i]:
            i += 1
          return str(n + i + 1)
    print('%d: failed to find effective line in und for %s'%(
       lineno(), u))
    return u.line

def parse_metrics(fname):
    f = open(fname)
    s = f.read().split('\n')
    f.close()
    results = {'class':[],'function':[], 'file':[], 'struct'
       :[]}

    currentFun = None
    '''
    while 1:
      if s[0] == 'All Entity Metrics:':
        break
      s.pop(0)
    s.pop(0)
```

```
s.pop(0)
print(s[0])
'''
for i in range(len(s)):
  if s[i] == 'All␣Entity␣Metrics:':
    break
i += 2
while i < len(s):
  l = s[i]
  if l == '' and i == len(s) - 1:
    break
  l = l.strip('␣').strip('\t')
  what = l[:l.find(':')-1]
  m = re.match('(.+?)␣:␣(.*?)␣:␣(.*?)␣:␣(.*?)␣:␣*(?:\\[
      File:␣(.*?)␣Line:␣(\\d+)\\])?', l)
  if what in ['File']:
    m = re.match('File␣:␣␣:␣([^:]+?)␣:␣␣:', l)
    fFile = m.group(1)
    currentFun = Function(fFile, -1, fFile)
    results['file'].append( currentFun )
  elif what in ['Function', 'Function␣Template', 'Public␣
      Function',
  'Public␣Const␣Function', 'Public␣Virtual␣Const␣Function
      ',
  'Public␣Virtual␣Function', 'Static␣Function', '
      Protected␣Const␣Function',
  'Protected␣Function', 'Private␣Function', 'Public␣
      Static␣Function',
  'Public␣Function␣Template', 'Protected␣Static␣Function'
      ,
  'Protected␣Virtual␣Function', 'Explicit␣Public␣Function
      ',
  'Protected␣Virtual␣Const␣Function', 'Private␣Virtual␣
      Function',
  'Private␣Static␣Function', 'Private␣Const␣Function',
  'Private␣Virtual␣Const␣Function', 'Public␣Const␣
      Function␣Template',
  'Public␣Static␣Function␣Template', 'Static␣Function␣
      Template',
  'Protected␣Function␣Template']:
    m = re.match('(.+?)␣:␣(.*?)␣:␣(.+?)␣:␣(.*?)␣:␣*(?:\\[
        File:␣(.*?)␣Line:␣(\\d+)\\])?', l)
    fType, fRet, fName, fArgs, fFile, fLine = m.groups()
    if fLine:
      fLine = str(int(fLine)+1) # start counting on 1
    currentFun = Function(fName, fLine, fFile)
    results['function'].append( currentFun )
  elif what in ['Abstract␣Class', 'Class␣Template', '
```

```
    Abstract␣Class␣Template',
'Private␣Class', 'Class', 'Public␣Class', 'Protected␣
    Class',
'Public␣Abstract␣Class']:
  m = re.match('(.+?)␣:␣␣:␣(.+?)␣:␣␣:␣+(?:\\[File:␣
      (.*?)␣Line:␣(\\d+)\\])', l)
  if m:
    fType, fName, fFile, fLine = m.groups()
  else:
    m = re.match('(.+?)␣:␣␣:␣(.+?)␣:␣␣:␣*$', l)
    fType, fName = m.groups()
    fFile = None
    fLine = None
    print('WARNING:␣no␣filename␣for␣%s'%(fName))
  currentFun = Function(fName, fLine, fFile)
  results['class'].append( currentFun )
elif what in ['Abstract␣Struct', 'Abstract␣Struct␣
    Template',
'Public␣Struct␣Template', 'Struct', 'Struct␣Template',
    'Public␣Struct',
'Protected␣Struct', 'Private␣Struct', 'Union', 'Public␣
    Union', 'Private␣Union']:
  m = re.match('(.+?)␣:␣␣:␣(.+?)␣:␣␣:␣*(?:\\[File:␣
      (.*?)␣Line:␣(\\d+)\\])?', l)
  fType, fName, fFile, fLine = m.groups()
  if fLine:
    fLine = str(int(fLine)+1) # start counting on 1
  currentFun = Function(fName, fLine, fFile)
  results['struct'].append( currentFun )
else:
  print('%d:␣failed␣to␣parse␣line\n%s'%(lineno(), l))
  exit(1)
if not m:
  print('%d:␣failed␣to␣parse␣line\n%s'%(lineno(), l))
  exit(1)
i += 1
while i < len(s) and s[i]:
  l = s[i]
  l = l.strip('␣').strip('\t')
  m = re.match('([^:]+):␣(\\d+(?:\\.\\d+)?)', l)
  if not m:
    print('%d:␣failed␣to␣parse␣line\n%s'%(lineno(), l))
    exit(1)
  name, val = m.groups()
  currentFun.metrics[name] = val
  i+= 1
if not s[i]:
  currentFun = None
```

```
      i += 1

    for y in results:
      for x in results[y]:
        x.line = get_real_line_number_from_und(x)
    #exit(1)

    return results

def group_metrics(metrics):
    name2fun = {
      'CountLinePreprocessor': sum,
      'CountLine': sum,
      'Cyclomatic': sum,
      'CountLineCode': sum,
      'CountOutput': sum,
      'MaxEssentialKnots': max,
      'CountInput': sum,
      'AltCountLineBlank': sum,
      'CountLineCodeExe': sum,
      'CountLineBlank': sum,
      'CountStmtDecl': sum,
      'AltCountLineCode': sum,
      'CountPath': sum,
      'CountLineComment':sum,
      'CountSemicolon': sum,
      'CyclomaticStrict': sum,
      'CountLineInactive': sum,
      'AltCountLineComment': sum,
      'CountLineCodeDecl': sum,
      'MinEssentialKnots': min,
      'CountStmtExe': sum,
      'CountStmt': sum,
      'Essential': sum,
      'Knots': sum,
      'MaxNesting': max,
      'RatioCommentToCode': lambda x: sum(x)/float(len(x)),
      'CyclomaticModified': sum,
      'CountStmtEmpty': sum,
    }
    results = {}
    for x in metrics:
      results[x] = name2fun[x](
        [ (float if '.' in y else int)(y) for y in metrics[x]
            ]
      )
    return results
```

```
calls, metrics, classes, structs = None,None,None,None
def main():
  global calls, metrics, classes, structs
  calls = parse_calls('funtrace')
  metrics = parse_metrics('und')
  classes = metrics['class']
  structs = metrics['struct']
  metrics = metrics['function'] #+ metrics['class']
# raise RuntimeError('Interactive Mode')
  for c in calls:
    if '~' in c.name: continue # ignore destructors
    if not c.filename or \
    'home' not in c.filename:
      continue
    shortc = get_very_short_fun_name(c.name)

    foundClasses = []
    #if 'CryptoPP::AllocatorWithCleanup' in c.name:
      #print(c.name, c.filename, c.line)
      #print('!',shortc)
      #exit(1)
    for m in classes + structs:
      shortm = get_very_short_fun_name(m.name)
      #if 'CryptoPP::AllocatorWithCleanup' in c.name:
        #print(shortm)
      if re.search('^%s::'%(shortm), shortc) or shortm ==
         shortc:
        foundClasses.append(m)
        #if 'CryptoPP::AllocatorWithCleanup' in c.name:
          #print('-',shortm)
          #print('+',m.name)
    if 0 == len(foundClasses):
      print('no␣classes␣found␣for:\n%s'%(c))
      #exit(1)
    elif 1 == len(foundClasses):
      foundClasses[0].mark = True
    else:
      print('more␣than␣a␣single␣class␣matched:\n%s'%(c))
      for x in foundClasses:
        x.mark = True
        print(x)
    #if 'CryptoPP::AllocatorWithCleanup' in c.name:
      #print(foundClasses)
      #exit(1)
    '''
    elif 1 < len(foundClasses):
      print('found more than a single matching class for:\n
```

```
            %s'%(c))
          print(foundClasses)
          exit(1)
        '''
        found = []
#       if 'DOMLSParserImpl::startElement' in c.name:
#         print('PISS',[x for x in metrics if 'DOMLSParserImpl
    ::startElement' in x.name])
        for m in metrics:
#         if 'DOMLSParserImpl::startElement' in c.name and '
    DOMLSParserImpl::startElement' in m.name:
#           print(m, )
#           print('-'*32)
          if c.filename == m.filename:
#           print(m.name)
            f = open(c.filename)
            src = f.read().split('\n')
            f.close()
            small, large = int(c.line), int(m.line)
            small, large = min(small, large), max(small, large)
            small -= 1
            large -= 1
            i = small
#           if 'DOMLSParserImpl::startElement' in c.name:
#             print(small, large)
            while i <= large:
              s = src[i]
              for x in ' \r\t':
                s = s.replace(x,'')
              if not s:
                src.pop(i)
                large -= 1
              else:
                i += 1
            if large - small == 0:
              found = [m]
#             if 'DOMLSParserImpl::startElement' in c.name:
#               print('!',m)
              break
            if large - small <= 3:
              found.append(m)
            '''
            if abs(int(c.line) - int(m.line)) == 0:
              found = [m]
              break
            if abs(int(c.line) - int(m.line)) <= 3:
              found.append(m)
            '''
```

```
    '''
    l = []
    shortc = get_very_short_fun_name(c.name)
    for m in foundClasses:
      shortm = get_very_short_fun_name(m.name)
      altm = shortm
      if '::' in shortm:
        altm = shortm + '::' + shortm.split('::')[-1]
      if ((shortc in [shortm, altm])
      or (abs(int(c.line) - int(m.line)) == 0)):
        l.append(m)
    foundClasses = l
    '''
    l = []
    '''
    if('non-virtual thunk to CryptoPP::Rijndael::Enc::
       AdvancedProcessBlocks'
    in c.name):
      print(found, shortc, c.filename, c.line)
      for m in metrics:
        if 'CryptoPP::Rijndael::Enc::AdvancedProcessBlocks'
            in m.name:
          shortm = get_very_short_fun_name(m.name)
          print(m)
          print(shortm)
      exit(1)
    #'''
    for m in found:
      shortm = get_very_short_fun_name(m.name)
      altm = shortm
      if '::' in shortm:
        altm = shortm + '::' + shortm.split('::')[-1]
      '''
      if 'SecureWipeBuffer' in c.name:
        print('?',c)
        print('?',m)
        print('!',shortc)
        print('!',shortm)
        print('!',altm)
      #'''
      if ((shortc in [shortm, altm])
      or (abs(int(c.line) - int(m.line)) == 0)):
        l.append(m)
    found = l
    #'''
    if len(found) == 0:
      if re.search('^
          __static_initialization_and_destruction_0', shortc
```

```
            ):
              pass
            elif('::operator=(' in c.name
            and re.search('::operator=\\(%s␣const&\\)$'%(c.name[:
                c.name.find('::operator=(')]), c.name)):
              pass # TODO this is implicit, we still have to mark
                  the class somefuckinghow
            elif '::' in shortc and shortc.split('::')[-1] ==
                shortc.split('::')[-2]:
              #print('constructor:\n%s'%(c))
              pass # TODO this is a constructor, we have to
                 somehow mark its class
            else:
              print('no␣matches:')
              print(c)
#             sys.exit(1)
          elif len(found) == 1:
            found[0].mark = 1
          elif len(found) > 1:
            print('%d:␣more␣than␣a␣single␣match:'%(lineno()))
            print(c.name, c.line)
            for x in found:
              print('\t'+str(x))
            print('-'*10)
            for x in found:
              x.mark = 1
#           exit(1)
          '''
          if(len(found) == 0 and len(foundClasses) != 1 and c.
              name[:2] != '__'
          and not [ 1 for x in ['g_actualMac', 'BlockCipherImpl',
              'DefaultObjectFactory'] if x in c.name]):
            exit(1)
          '''
          '''
          if len(found) == 0:
            if 'home' in c.filename:
              print(c)
              print('-'*10)
          '''
        print('-'*79)
        results = {}
        for m in metrics:
          if m.mark:
            #print(m)
            for x in m.metrics:
              if x not in results:
                results[x] = []
```

```
            results[x].append(m.metrics[x])
    results = group_metrics(results)
    for x in results:
      print('%30s\t%s'%(x, results[x]))


if __name__ == '__main__':
# main()
    import cProfile
    cProfile.run('main()','profiling')

import sys,os

def debug_exception(type, value, traceback):
    # Restore redirected standard I/O
    sys.stdin = sys.__stdin__
    sys.stdout = sys.__stdout__
    sys.stderr = sys.__stderr__

    # Kick the interpreter into interactive mode and call the
        original
    # exception handler.
    os.environ['PYTHONINSPECT'] = '1'
    sys.__excepthook__(type, value, traceback)

sys.excepthook = debug_exception

#!/usr/bin/env python
import os
import sys
import subprocess as sp
from cmds import cmds

gResultsDir = 'results'

'''
ok
to be checked
yet to be run in workdamnit.py
    cjpeg
    djpeg
    cryptopp_aes
    cryptopp_des
    cryptopp_rsa
    cryptopp_signverify
    gzip
    jpg2000enc
    jpg2000dec
    links
```

```
  mpeg1
  mibenchblowfish
  mibenchrijndael
  sha
  h264dec
  h264enc
  mpeg2decode
  mpeg2encode
yet to be run in go.py
  xercessax
  xercesdom
  sablot
  mpeg2decode_cpp
  h263enc
  h263dec
'''

def run(bin, args):
  if type(args) == str:
    args = args.split(' ')
  print('* running %s %s'%(bin, ' '.join(args)))
  p = sp.Popen([bin] + args, stderr=sp.PIPE, stdout=sp.PIPE
    )
  output = p.communicate()
  p.wait()
  return output

def nm(pr):
  return run('nm', ['-p', '-a', '-C', '-l', os.path.join('
    bin', pr['name'], pr['bin'])])

def funtrace(pr):
  return run('./callgrind.py', pr['name'])

def und(pr):
  return run('./get_metrics.sh', os.path.join('/home/luis/
    metricas/valgrind', pr['name'], pr['name'] + '.udb'))

#nm        = False
funtrace = False
und       = False

def main():
  global cmds
  if not os.path.isdir(gResultsDir):
    os.mkdir(gResultsDir)
  if len(sys.argv) > 1:
    selected = sys.argv[1:]
```

```
  else:
    selected = [x['name'] for x in cmds]
  for x in selected:
    if x not in [y['name'] for y in cmds]:
      print('failed to find %s'%(x))
      sys.exit(1)
  cmds = [ x for x in cmds if x['name'] in selected ]

  for pr in cmds:
    print(pr['name'])
    prDir = os.path.join(gResultsDir, pr['name'])
    if not os.path.isdir(prDir):
      os.mkdir(prDir)

    if nm:
      stdout, stderr = nm(pr)
      if stderr:
        print(stderr)
        exit(1)
      f = open(os.path.join(prDir, 'nmdata'), 'wt')
      f.write(stdout)
      f.close()

    if funtrace:
      stdout, stderr = funtrace(pr)
      if stdout:
        print(stdout)
      if stderr:
        print(stderr)
        exit(1)

    if und:
      stdout, stderr = und(pr)
      if stderr:
        print(stderr)
        exit(1)
      f = open(os.path.join(prDir, 'und'), 'wt')
      f.write(stdout)
      f.close()
  print('success')
if __name__ == '__main__': main()
```

```
#!/usr/bin/python
import subprocess as sp
import sys
from multiprocessing import Pool
def run(bin, args):
        if type(args) == str:
                args = args.split(' ')
```

```
        print('* running %s %s'%(bin, ' '.join(args)))
        p = sp.Popen([bin] + args, stderr=sp.PIPE, stdout=
           sp.PIPE)
        output = p.communicate()
        p.wait()
        return output
def run_single_arg(prargs):
        out = run(prargs[0], prargs[1])
        print(out[0])
        print(out[1])
        return out

def main():
        pr = sys.argv[1]
        args = sys.argv[2:]
        p = Pool(len(args))
        p.map(run_single_arg, [ (pr, x) for x in args ])
if __name__ == '__main__': main()
```

```
#!/usr/bin/env python
import os
from sys import exit
import re
from glob import glob
import subprocess as sp
#import cPickle as pickle

def compile(tail=[]):
  if type(tail) == str:
    tail = [tail]
  return run('g++', [
    '-DDOM', '-ovalidate_dom',
    '-m32', '-march=i686', '-g', '-static', '-O0',
    'DOMPrint.cpp', 'DOMPrintFilter.cpp', 'load-grammar-dom
       .cpp', 'main.cpp',
    'SAXPrintHandlers.cpp', 'DOMPrintErrorHandler.cpp', '
       DOMTreeErrorReporter.cpp',
    'load-grammar-sax.cpp', 'SAXPrint.cpp',
    '-I../../src', '-L../../src/.libs',
    ] + tail)[1]

def find_files(root):
  files = []
  if root[-1]!='*':
    root=os.path.join(root,'*')
  l=glob(root)
  while l:
    f=l.pop(0)
    if os.path.isdir(f):
```

```
      l += glob(os.path.join(f,'*'))
    else:
      files.append(f)
  src = [x for x in files if x[x.rfind('.'):] in ['.c','.cc
    ','.cpp']]
  obj = [x for x in files if x[x.rfind('.'):] in ['.o']]
  return (src, obj)

def run(bin, args, shell=False, stdin=None):
  if type(args) == str:
    args = args.split(' ')
  print('* running %s %s'%(bin, ' '.join(args)))
  if stdin:
    p = sp.Popen([bin] + args, shell=shell, stderr=sp.PIPE,
        stdout=sp.PIPE, stdin=sp.PIPE)
    output = p.communicate(stdin)
  else:
    p = sp.Popen([bin] + args, shell=shell, stderr=sp.PIPE,
        stdout=sp.PIPE)
    output = p.communicate()
  p.wait()
  return output

def nm(fname):
  return run('nm', ['-p', '-a', '-C', '-l', fname])

def get_functions_from_object(o):
  results = []
  lines = nm(o)[0].split('\n')
  for l in lines:
    if not l: continue
    m = re.match('^(.+)\\s(.)\\s([^\t]*)(?:\t(.+))?$', l)
    addr, typ, name, fname = m.groups()
    if typ in 'TRV' and name not in results:
      results.append(name)
#   if 'xercesc_3_1::SimpleContentModel::
   getContentLeafNameTypeVector' in l:
#     print(l, name, o)
  return results

def find_deps(tail=[]):
  lines = compile(tail).split('\n')
  undefined = []
  for l in lines:
#   m = re.match('.*:\\d+: undefined reference to `(.*)\'',
    l)
    m = re.match('.*: undefined reference to `(.*)\'', l)
    if m:
```

```
        name = m.group(1)
#       if '(' in name:
#         name = name[:name.find('(')]
#       if '::' in name:
#         name = name.split('::')[-1]
#       print(name)
        undefined.append(name)
    return undefined

def map_functions_to_obj(obj):
    d = {}
    if type(obj) != list:
        obj = [obj]
    for o in obj:
        for fun in get_functions_from_object(o):
            if fun not in d:
                d[fun] = [o]
            else:
                d[fun].append(o)
    return d

def main():
    src, obj = find_files('../../*')
    function2obj = map_functions_to_obj(obj)

    try:
        f = open('linkWith', 'rt')
        linkWith = [x for x in f.read()[1:-1].split('"␣"') if x
            ]
        f.close()
    except IOError:
        linkWith  = []
    linkDict = {}
    for x in linkWith:
        linkDict[x] = None

    while 1:
        undefined = find_deps(linkWith)
        l = []
        for x in undefined:
            if x in function2obj and 1 == len(function2obj[x]):
                o = function2obj[x][0]
                if o not in linkDict:
                    l.append(o)
                    linkWith.append(o)
                    linkDict[o] = None
        if not l:
            break
```

```
      print(l)

   print('-'*40)
   print(linkWith)
   f = open('linkWith','wt')
   f.write('"%s"'%('" "'.join(linkWith)))
   f.close()
main()

#!/usr/bin/env python
import os

top = '/home/luis/metricas/metricas/profiling/valgrind'
resources = '/home/luis/SPECjvm2008/resources'
bindir = os.path.join(top,'bin')
tmpdir = os.path.join(top,'tmp')
cmds = [
{
'name':'sablot',
'bin':'sablot',
'args':[
  [os.path.join(resources,'xml.transform'),'0'],
  [os.path.join(resources,'xml.transform'),'1'],
  [os.path.join(resources,'xml.transform'),'2'],
  [os.path.join(resources,'xml.transform'),'3'],
  [os.path.join(resources,'xml.transform'),'4'],
  [os.path.join(resources,'xml.transform'),'5'],
  [os.path.join(resources,'xml.transform'),'6'],
  [os.path.join(resources,'xml.transform'),'7'],
  [os.path.join(resources,'xml.transform'),'8'],
  [os.path.join(resources,'xml.transform'),'9'],
  ]
},
{
'name':'cryptopp_aes',
'bin':'test_aes_des',
'args':[
  ['9f9f90dbe3e5ee1218c86b8839db1995','0',os.path.join(
     resources,'crypto/fredmans21.txt'),os.path.join(tmpdir
     ,'cryptopp_aes/0/output')],
  ['9f9f90dbe3e5ee1218c86b8839db1995','0',os.path.join(
     resources,'crypto/track3.mp3'),os.path.join(tmpdir,'
     cryptopp_aes/1/output')],
  ]
},
{
'name':'cryptopp_des',
'bin':'test_des_ede',
'args':[
```

```
  ['omega',os.path.join(resources,'crypto/random96.dat')],
  ['omega',os.path.join(resources,'crypto/random65536.dat')
     ],
  ]
},
{
'name':'cryptopp_rsa',
'bin':'test_rsa',
'args':[
  [os.path.join(bindir,'cryptopp_rsa/rsa/rsa_private_key'),
      os.path.join(bindir,'cryptopp_rsa/rsa/rsa_public_key'
     ), 'omega',os.path.join(resources,'crypto/random96.dat
     ')],
  [os.path.join(bindir,'cryptopp_rsa/rsa/rsa_private_key'),
      os.path.join(bindir,'cryptopp_rsa/rsa/rsa_public_key'
     ), 'omega',os.path.join(resources,'crypto/random65536.
     dat')],
  ]
},
{
'name':'cryptopp_signverify',
'bin':'test_signverify',
'args':[
  [os.path.join(bindir,'cryptopp_rsa/rsa/rsa_private_key'),
      os.path.join(bindir,'cryptopp_rsa/rsa/rsa_public_key'
     ), os.path.join(resources,'crypto/random1024.dat'), os
     .path.join(tmpdir, 'cryptopp_signverify/0/output')],
  [os.path.join(bindir,'cryptopp_rsa/rsa/rsa_private_key'),
      os.path.join(bindir,'cryptopp_rsa/rsa/rsa_public_key'
     ), os.path.join(resources,'crypto/random1048576.dat'),
      os.path.join(tmpdir, 'cryptopp_signverify/1/output')
     ],
  ]
},
{
'name':'xercessax',
'bin':'validate_sax',
'args':[
  [os.path.join(resources,'xml.validation')],
  ]
},
{
'name':'xercesdom',
'bin':'validate_dom',
'args':[
  [os.path.join(resources,'xml.validation')],
  ]
},
```

```
{'name':'gzip',
'bin':'gzip',
'args':[
  [os.path.join(tmpdir,'gzip/0/205.tar')],
  [os.path.join(tmpdir,'gzip/1/208.tar')],
  [os.path.join(tmpdir,'gzip/2/210.tar')],
  ['-d',os.path.join(tmpdir,'gzip/0/205.tar.gz')],
  ['-d',os.path.join(tmpdir,'gzip/1/208.tar.gz')],
  ['-d',os.path.join(tmpdir,'gzip/2/210.tar.gz')],
  ]
},
{'name':'cjpeg',
'bin':'cjpeg',
'args':[
  ['-dct','int','-quality','90','-outfile','/dev/null',os.
     path.join(tmpdir, 'cjpeg/0/input_base_4CIF.ppm')],
  ]
},
{'name':'djpeg',
'bin':'djpeg',
'args':[
  ['-dct','int','-ppm','-outfile','/dev/null',os.path.join(
     tmpdir,'djpeg/0/input_base_4CIF_96bps.jpg')],
  ]
},
{'name':'mpeg2decode',
'bin':'mpeg2decode',
'args':[
  ['-b', os.path.join(tmpdir,'mpeg2decode/0/
     input_base_4CIF_96bps.mpg'), '-o3', os.path.join(
     tmpdir, 'mpeg2decode/0/output')],
]
},
{'name':'mpeg2encode',
'bin':'mpeg2encode',
'args':[
  [os.path.join(tmpdir,'mpeg2encode/0/
     input_base_4CIF_96bps_15.par'),'/dev/null'],
]
},
{'name':'jpg2000enc',
'bin':'jpg2000enc',
'args':[
  ['-f',os.path.join(tmpdir,'jpg2000enc/0/input_base_4CIF.
     ppm'),'-F',os.path.join(tmpdir,'jpg2000enc/0/output.
     jp2'),'-T','jp2','-O','rate=0.010416667']
]
},
```

```
{'name':'jpg2000dec',
'bin':'jpg2000dec',
'args':[
  ['-f',os.path.join(tmpdir,'jpg2000dec/0/
     input_base_4CIF_96bps.jp2'),'-F',os.path.join(tmpdir,'
     jpg2000dec/0/output.ppm'),'-T','pnm']
]
},
{'name':'h263enc',
'bin':'tmn',
'args':[
  ['-x', '4', '-a', '0', '-b', '8', '-s', '15', '-G', '-R',
      '30.00', '-r', '3508000', '-S', '3', '-Z', '30.0', '-
     O', '0', '-i', os.path.join(tmpdir, 'h263enc/0/
     input_base_4CIF_0to8.yuv'), '-o', '/dev/null', '-B',
     os.path.join(tmpdir, 'h263enc/0/
     output_base_4CIF_96bps_15.263')]
]
},
{'name':'h263dec',
'bin':'tmndec',
'args':[
  ['-o3',os.path.join(tmpdir, 'h263dec/0/
     input_base_4CIF_96bps.263'),'out_h263dec']
]
},
{'name':'h264dec',
'bin':'ldecod.exe',
'args':[
  ['input_base_4CIF_96bps_decoder.cfg']
]
},
{'name':'h264enc',
'bin':'lencod.exe',
'args':[
  ['-d',os.path.join(tmpdir, 'h264enc/0/
     input_base_4CIF_96bps_encoder.cfg')]
]
},
{'name':'mibenchblowfish',
'bin':'bf',
'args':[
  ['e',os.path.join(tmpdir, 'mibenchblowfish/0/fredmans21.
     txt'),'/dev/null','9f9f90dbe3e5ee1218c86b8839db1995'],
  ['e',os.path.join(tmpdir, 'mibenchblowfish/1/track3.mp3')
     ,'/dev/null','9f9f90dbe3e5ee1218c86b8839db1995'],
  ['d',os.path.join(tmpdir, 'mibenchblowfish/2/fredmans21.
     txt.encrypted'),'/dev/null','9
```

```
      f9f90dbe3e5ee1218c86b8839db1995'],
  ['d',os.path.join(tmpdir, 'mibenchblowfish/3/track3.mp3.
      encrypted'),'/dev/null','9
      f9f90dbe3e5ee1218c86b8839db1995'],
  # the two following benchmarks were not used in the
      simulator, however we use them here as the other four
      end too fast for gprof to measure them
#   ['e',os.path.join(tmpdir, 'mibenchblowfish/4/random128M
    '),'/dev/null','9f9f90dbe3e5ee1218c86b8839db1995'],
#   ['d',os.path.join(tmpdir, 'mibenchblowfish/5/random128M
    .encrypted'),'/dev/null','9
    f9f90dbe3e5ee1218c86b8839db1995'],
]
},
{'name':'mibenchrijndael',
'bin':'rijndael',
'args':[
  [os.path.join(tmpdir, 'mibenchrijndael/0/fredmans21.txt')
      ,'/dev/null', 'e', '9f9f90dbe3e5ee1218c86b8839db1995'
      ],
  [os.path.join(tmpdir, 'mibenchrijndael/0/track3.mp3'),'/
      dev/null','e', '9f9f90dbe3e5ee1218c86b8839db1995'],
  [os.path.join(tmpdir, 'mibenchrijndael/0/fredmans21.txt.
      encrypted'),'/dev/null', 'd', '9
      f9f90dbe3e5ee1218c86b8839db1995'],
  [os.path.join(tmpdir, 'mibenchrijndael/0/track3.mp3.
      encrypted'),'/dev/null','d', '9
      f9f90dbe3e5ee1218c86b8839db1995'],
  # the two following benchmarks were not used in the
      simulator, however we use them here as the other four
      end too fast for gprof to measure them
#   [os.path.join(tmpdir, 'mibenchrijndael/0/random128M')
    ,'/dev/null','e', '9f9f90dbe3e5ee1218c86b8839db1995'],
#   [os.path.join(tmpdir, 'mibenchrijndael/0/random128M.
    encrypted'),'/dev/null','d', '9
    f9f90dbe3e5ee1218c86b8839db1995'],
]
},
{'name':'mpeg2decode_cpp',
'bin':'mpeg2decode_cpp',
'args':[
  ['-q','-b',os.path.join(tmpdir, 'mpeg2decode_cpp/0/
      input_base_4CIF_96bps.mpg'),'-o3',
  os.path.join(tmpdir,'mpeg2decode_cpp/0/out')],
]
},
{'name':'links',
'bin':'links',
```

```
'args':[
  ['-dump', os.path.join(tmpdir, 'links/0/index.html')],
  ]
},
{'name':'sha',
'bin':'sha',
'args':[
  [os.path.join(resources, 'crypto.signverify/validity.
     crypto.signverify.dat')],
  [os.path.join(tmpdir,'sha/0/random128M')],
]
},
{'name':'mpeg1',
'bin':'mpeg1',
'args':[
  [os.path.join(tmpdir, 'mpeg1/0/test.mp3')],
]
},
]
```

```
#!/usr/bin/env python
import os
import sys
import shutil
import subprocess as sp
from glob import glob
from cmds import *


gResultsDir = 'results'

if not os.path.isdir(gResultsDir):
  os.mkdir(gResultsDir)

def valgrind(bin, args):
  print(['/home/luis/src/valgrind/valgrind/install/bin/
     valgrind', '--tool=funtrace']
    + [bin] + args)
  p = sp.Popen(['/home/luis/src/valgrind/valgrind/install/
     bin/valgrind', '--tool=funtrace']
    + [bin] + args, stderr=sp.PIPE, stdout=sp.PIPE)

  '''
␣␣p␣=␣sp.Popen(['valgrind',␣'--tool=callgrind',␣'--trace-
   children=yes',
␣␣␣␣␣␣'--compress-pos=no',
␣␣␣␣␣␣'--callgrind-out-file=callgrind.out']
␣␣␣␣␣␣+␣[bin]␣+␣args,␣stderr=sp.PIPE,␣stdout=sp.PIPE)
  '''
␣␣
```

```
  output = p.communicate()[1]
  retval = p.wait()
  if 0 != retval:
    print(os.getcwd())
    print('RETVAL: %d'%(retval))
    sys.exit(1)
  fname = os.path.join(os.getcwd(), 'callgrind.out')
  f = open(fname, 'wt')
  f.write(output)
  f.close()
  return fname
def main():
  global top
  global resources
  global bindir
  global tmpdir
  global cmds

  selected = [
    'cryptopp_aes',
    'cryptopp_des',
    'cryptopp_rsa',
    'cryptopp_signverify',
    'sha',
  ]
  if len(sys.argv) > 1:
    selected = sys.argv[1:]
  cmds = [ x for x in cmds if x['name'] in selected ]

  if not os.path.isdir(tmpdir):
    os.mkdir(tmpdir)
  for cmd in cmds:
    cmd['rundirs'] = [ os.path.join(bindir,tmpdir,cmd['name
      '],str(i)) for i in range(len(cmd['args'])) ]
#   continue
    for dst in cmd['rundirs']:
      print(dst)
      if os.path.isdir(dst):
        shutil.rmtree(dst)
      shutil.copytree(os.path.join(bindir,cmd['name']), dst
        )

    shutil.copy(os.path.join(bindir,'xerces/validate_sax'),
      os.path.join(tmpdir,'xerces/0/xerces'))
    shutil.copy(os.path.join(bindir,'xerces/validate_dom'),
      os.path.join(tmpdir,'xerces/1/xerces'))

  for pr in cmds:
```

```
      prDir = os.path.join(gResultsDir, pr['name'])
      if not os.path.isdir(prDir):
        os.mkdir(prDir)
      for i in range(len(pr['args'])):
        prevDir = os.getcwd()
        os.chdir(os.path.join(tmpdir, pr['rundirs'][i]))
        fname = valgrind('./' + pr['bin'], pr['args'][i])
        os.chdir(prevDir)
        shutil.copyfile(fname, os.path.join(prDir, 'funtrace_
          %d'%(i)))
      buf = ''
      for i in range(len(pr['args'])):
        f = open(os.path.join(prDir, 'funtrace_%d'%(i)), 'rt'
          )
        buf += f.read() + '\n'
        f.close()
      f = open(os.path.join(prDir, 'funtrace'), 'wt')
      f.write(buf[:-1])
      f.close()
#       break
#     break
# os.shutil.rmtree(tmpdir)
main()
```

8.4 Scripts para Execução dos Benchmarks

```
#!/usr/bin/env python
import os
import sys
sys.path.append(sys.path[0]+'/..')
from m2s import multi2sim
sys.path.pop(-1)
#from shootout import get_shootout_benches

resources = 'crypto'
gRootDir = '/home/luis/src/m2s'

gPossibleRoots = ['/home/luis/src/m2s','/mnt/lfgmillani/
    newer','/mnt/lfgmillani/new']

gResourcesDir = os.path.join(os.path.join(gRootDir,'
    resources'), resources)

def getargsfromcmd(m2s,cmd):
  now = cmd
  bin=now[0]
  args=now[1]
  cpucache='cpucache_'+now[2]
  cpupipe='cpupipe_'+now[2]
```

```
  cmdhash = m2s.fix_run_args(bin,
        args,
        {'stdout':'/dev/stdout','stderr':'/dev/stderr',
          'cpu-cache':cpucache,
          'cpu-pipeline':cpupipe},
        [],'cpu.cfg')['cmdhash']
  return {'now':now,'bin':bin,'args':args,'cpucache':
     cpucache,'cpupipe':cpupipe,'cmdhash':cmdhash}

def getcmdhash(m2s, cmd):
  args=getargsfromcmd(m2s,cmd)
  return m2s.fix_run_args(args['bin'],args['args'], {'
     stdout':'/dev/stdout','stderr':'/dev/stderr',
                'cpu-cache':args['cpucache'],
                'cpu-pipeline':args['cpupipe']},
              [],'cpu.cfg')['cmdhash']


def getcmdhash2(m2s, cmd):
  now = cmd
  bin=now[0]
  args=now[1]
  cpucache='cpucache_'+now[2]
  cpupipe='cpupipe_'+now[2]
  cmdhash = m2s.fix_run_args(bin,
        args,
        {'stdout':'/dev/stdout','stderr':'/dev/stderr',
          'cpu-cache':cpucache,
          'cpu-pipeline':cpupipe},
        [],'cpu.cfg')['cmdhash']
  return cmdhash

def main():
  m2s = multi2sim(sys.path[0]+'/..')
  top = os.path.join(os.getcwd(),'../..')

  res = os.path.join(top,'resources')
  '''
  bin = os.path.join(top,'pr/cryptopp/test_des_ede')
  args = [res+'random96.dat']
  bin = os.path.join(top,'pr/cryptopp/test_aes_des')
  args = [res+'fredmans21.txt']
  bin = os.path.join(top,'pr/cryptopp/test_signverify')
  args = [res+'random1024.txt']
  bin = os.path.join(top,'pr/cryptopp/test_rsa')
  args = [res+'random96.txt']
  bin = os.path.join(top,'pr/sablot/sablot')
  args = [res,'5']
```

```
bin = os.path.join(top,'pr/sablot/sablot')
args = [res,'9']
bin = os.path.join(top,'pr/xerces/validate_sax')
args = [res]
bin = os.path.join(top,'pr/xerces/validate_dom')
args = [res]
'''
aes=os.path.join(top,'pr/cryptopp/test_aes_des')
des=os.path.join(top,'pr/cryptopp/test_des_ede')
rsa=os.path.join(top,'pr/cryptopp/test_rsa_encrypt')
sign=os.path.join(top,'pr/cryptopp/test_signverify')
sablot=os.path.join(top,'pr/sablot/sablot')
xercessax=os.path.join(top,'pr/xerces/validate_sax')
xercesdom=os.path.join(top,'pr/xerces/validate_dom')
mpeg1=os.path.join(top,'pr/mpeg1/mpeg1')
sha=os.path.join(top,'pr/sha/sha')
links=os.path.join(top,'pr/links/links')
gzip=os.path.join(top,'pr/gzip/gzip')
cjpeg=os.path.join(top,'pr/cjpeg/cjpeg')
djpeg=os.path.join(top,'pr/djpeg/djpeg')
mpeg2decode=os.path.join(top,'pr/mpeg2decode/mpeg2decode'
    )
mpeg2decode_cpp=os.path.join(top,'pr/mpeg2decode_cpp/
    mpeg2decode_cpp')
mpeg2encode=os.path.join(top,'pr/mpeg2encode/mpeg2encode'
    )
h264dec=os.path.join(top,'pr/h264dec/ldecod.exe')
h264enc=os.path.join(top,'pr/h264enc/lencod.exe')
h263dec=os.path.join(top,'pr/h263dec/tmndec')
h263enc=os.path.join(top,'pr/h263enc/tmn')
jpg2000dec=os.path.join(top,'pr/jpg2000dec/jpg2000dec')
jpg2000enc=os.path.join(top,'pr/jpg2000enc/jpg2000enc')
mibenchblowfish=os.path.join(top,'pr/mibenchblowfish/bf')
mibenchpgp=os.path.join(top,'pr/mibenchpgp/pgp')
mibenchrijndael=os.path.join(top,'pr/mibenchrijndael/
    rijndael')
python=os.path.join(top,'pr/python/install/bin/python3')
l = [
  # 0
  [aes,['9f9f90dbe3e5ee1218c86b8839db1995','0',res+'/
      crypto/fredmans21.txt','/dev/null'],'aes_fredmans21.
      txt'],
  # 1
  [aes,['9f9f90dbe3e5ee1218c86b8839db1995','0',res+'/
      crypto/track3.mp3','/dev/null'],'aes_track3.mp3'],
  # 2
  [des,['omega',res+'/crypto/random96.dat'],'des_random96
      .dat'],
```

```
# 3
[rsa,[os.path.join(os.path.dirname(rsa),'rsa/
   rsa_private_key'),
       os.path.join(os.path.dirname(rsa),'rsa/
          rsa_public_key'),
       'omega',res+'/crypto/random96.dat'],'rsa_random96
          .dat'],
# 4
[rsa,[os.path.join(os.path.dirname(rsa),'rsa/
   rsa_private_key'),
       os.path.join(os.path.dirname(rsa),'rsa/
          rsa_public_key'),
       'omega',res+'/crypto/random65536.dat'],'
          rsa_random65536.dat'],
# 5
[sign,[os.path.join(os.path.dirname(rsa),'rsa/
   rsa_private_key'),
        os.path.join(os.path.dirname(rsa),'rsa/
           rsa_public_key'),
        res+'/crypto/random1024.dat','
           delme_sign_random1024.dat'],'sign_random1024.
           dat'],
# 6
[sign,[os.path.join(os.path.dirname(rsa),'rsa/
   rsa_private_key'),
        os.path.join(os.path.dirname(rsa),'rsa/
           rsa_public_key'),
        res+'/crypto/random65536.dat','
           delme_sign_random65536.dat'],'
           sign_random65536.dat'],
# 7
[sign,[os.path.join(os.path.dirname(rsa),'rsa/
   rsa_private_key'),
        os.path.join(os.path.dirname(rsa),'rsa/
           rsa_public_key'),
        res+'/crypto/random1048576.dat','
           delme_sign_random1048576.dat'],'
           sign_random1048576.dat'],
# 8
[sablot,[res+'/xml.transform','5'],'sablot_5'],
# 9
[sablot,[res+'/xml.transform','8'],'sablot_8'],
# 10
[sablot,[res+'/xml.transform','9'],'sablot_9'],
# 11
[xercessax,[res+'/xml.validation'],'xerces_sax'],
# 12
[xercesdom,[res+'/xml.validation'],'xerces_dom'],
```

```
# 13
[sablot,[res+'/xml.transform','0'],'sablot_0'],
# 14
[sablot,[res+'/xml.transform','1'],'sablot_1'],
# 15
[sablot,[res+'/xml.transform','2'],'sablot_2'],
# 16
[sablot,[res+'/xml.transform','3'],'sablot_3'],
# 17
[sablot,[res+'/xml.transform','4'],'sablot_4'],
# 18
[sablot,[res+'/xml.transform','6'],'sablot_6'],
# 19
[sablot,[res+'/xml.transform','7'],'sablot_7'],
# 20
[mpeg1,[os.path.dirname(mpeg1)+'/test.mp3'],'mpeg1_0'],
# 21
[sha,[res+'/crypto.signverify/validity.crypto.
    signverify.dat'],'sha'],
# 22
[des,['omega',res+'/crypto/random65536.dat'],'
    des_random65536.dat'],
# 23
[links,['-dump',os.path.join(os.path.dirname(links), '
    index.html')],'links_0'],
# 24
[mpeg1,[os.path.dirname(mpeg1)+'/1m.mp3'],'mpeg1_1m'],
# 25
[mpeg1,[os.path.dirname(mpeg1)+'/100ms.mp3'],'
    mpeg1_100ms'],
# 26
[mpeg1,[os.path.dirname(mpeg1)+'/1s.mp3'],'mpeg1_1s'],
# 27
[gzip,[os.path.dirname(gzip)+'/210.tar'],'gzip_compress
    '],
# 28
[gzip,['-d',os.path.dirname(gzip)+'/210.tar.gz'],'
    gzip_decompress'],
# 29
[gzip,[os.path.dirname(gzip)+'/205.tar'],'gzip_compress
    '],
# 30
[gzip,['-d',os.path.dirname(gzip)+'/205.tar.gz'],'
    gzip_decompress'],
# 31
[gzip,[os.path.dirname(gzip)+'/202.tar'],'gzip_compress
    '],
# 32
```

```
[gzip,['-d',os.path.dirname(gzip)+'/202.tar.gz'],'
    gzip_decompress'],
# 33
[gzip,[os.path.dirname(gzip)+'/208.tar'],'gzip_compress
    '],
# 34
[gzip,['-d',os.path.dirname(gzip)+'/208.tar.gz'],'
    gzip_decompress'],
# 35
[gzip,[os.path.dirname(gzip)+'/209.tar'],'gzip_compress
    '],
# 36
[gzip,['-d',os.path.dirname(gzip)+'/209.tar.gz'],'
    gzip_decompress'],
# 37
[gzip,[os.path.dirname(gzip)+'/211.tar'],'gzip_compress
    '],
# 38
[gzip,['-d',os.path.dirname(gzip)+'/211.tar.gz'],'
    gzip_decompress'],
# 39
[gzip,[os.path.dirname(gzip)+'/213x.tar'],'
    gzip_compress'],
# 40
[gzip,['-d',os.path.dirname(gzip)+'/213x.tar.gz'],'
    gzip_decompress'],
# 41
[gzip,[os.path.dirname(gzip)+'/228.tar'],'gzip_compress
    '],
# 42
[gzip,['-d',os.path.dirname(gzip)+'/228.tar.gz'],'
    gzip_decompress'],
# 43
[gzip,[os.path.dirname(gzip)+'/239.tar'],'gzip_compress
    '],
# 44
[gzip,['-d',os.path.dirname(gzip)+'/239.tar.gz'],'
    gzip_decompress'],
# 45
[gzip,[os.path.dirname(gzip)+'/misc.tar'],'
    gzip_compress'],
# 46
[gzip,['-d',os.path.dirname(gzip)+'/misc.tar.gz'],'
    gzip_decompress'],
# 47
[cjpeg,['-dct','int','-quality','90','-outfile','/dev/
    null',os.path.dirname(cjpeg)+'/input_base_4CIF.ppm'
    ],'cjpeg_input_base_4CIF'],
```

```
# 48
[djpeg,['-dct','int','-ppm','-outfile','/dev/null',os.
    path.dirname(djpeg)+'/input_base_4CIF_96bps.jpg'],'
    djpeg_input_base_4CIF_96bps'],
# 49
[mpeg2decode,['-b',os.path.dirname(mpeg2decode)+'/
    input_base_4CIF_96bps.mpg','-o3',os.path.dirname(
    mpeg2decode)+'/output'],'mpeg2decode'],
# 50
[mpeg2encode,[os.path.dirname(mpeg2encode)+'/
    input_base_4CIF_96bps.par','/dev/null'],'mpeg2encode
    '],
# 51
[h264dec,['input_base_4CIF_96bps_decoder.cfg'],'h264dec
    '],
# 52
[h264enc,['-d',os.path.dirname(h264enc)+'/
    input_base_4CIF_96bps_encoder.cfg'],'h264enc'],
# 53
[h263dec,['-o3',os.path.dirname(h263dec)+'/
    input_base_4CIF_96bps.263','out_h263dec'],'h263dec_0
    '],
# 54
[h263enc,['-x', '4', '-a', '0', '-b', '8', '-s', '15',
    '-G', '-R', '30.00', '-r', '3508000', '-S', '3', '-Z
    ', '30.0', '-O', '0', '-i', os.path.dirname(h263enc)
    +'/input_base_4CIF_0to8.yuv', '-o', '/dev/null', '-B
    ', 'output_base_4CIF_96bps_15.263'],'h263_enc'],
# 55
[h264enc,['-d',os.path.dirname(h264enc)+'/ycsd.cfg'],'
    h264enc_1'],
# 56
[jpg2000dec,['-f',os.path.dirname(jpg2000dec)+'/
    input_base_4CIF_96bps.jp2','-F',os.path.dirname(
    jpg2000dec)+'/output.ppm','-T','pnm'],'jpg2000dec'],
# 57
[jpg2000enc,['-f',os.path.dirname(jpg2000enc)+'/
    input_base_4CIF_96bps.ppm','-F',os.path.dirname(
    jpg2000enc)+'/output.jp2','-T','jp2','-O','rate
    =0.010416667'],'jpg2000enc'],
# 58
[mibenchblowfish,['e',res+'/crypto/fredmans21.txt','/
    dev/null','9f9f90dbe3e5ee1218c86b8839db1995'],'
    mibenchblowfish_enc_fredmans21.txt'],
# 59
[mibenchblowfish,['e',res+'/crypto/track3.mp3','/dev/
    null','9f9f90dbe3e5ee1218c86b8839db1995'],'
    mibenchblowfish_enc_track3.mp3'],
```

```
# 60
[mibenchblowfish,['d',os.path.dirname(mibenchblowfish)+
    '/fredmans21.txt.encrypted','/dev/null','9
    f9f90dbe3e5ee1218c86b8839db1995'],'
    mibenchblowfish_dec_fredmans21.txt'],
# 61
[mibenchblowfish,['d',os.path.dirname(mibenchblowfish)+
    '/track3.mp3.encrypted','/dev/null','9
    f9f90dbe3e5ee1218c86b8839db1995'],'
    mibenchblowfish_dec_track3.mp3'],
# 62
[mibenchpgp,['-es',res+'/crypto/random1024.dat','
    otheruser'],'mibenchpgp_random1024.dat'],
# 63
[mibenchpgp,['-es',res+'/crypto/random65536.dat','
    otheruser'],'mibenchpgp_random65536.dat'],
# 64
[mibenchpgp,['-es',res+'/crypto/random1048576.dat','
    otheruser'],'mibenchpgp_random1048576.dat'],
# 65
[mibenchrijndael,[res+'/crypto/fredmans21.txt','/dev/
    null','e','9f9f90dbe3e5ee1218c86b8839db1995'],'
    mibenchrijndael_enc_fredmans21.txt'],
# 66
[mibenchrijndael,[res+'/crypto/track3.mp3','/dev/null',
    'e','9f9f90dbe3e5ee1218c86b8839db1995'],'
    mibenchrijndael_enc_track3.mp3'],
# 67
[mibenchrijndael,[os.path.dirname(mibenchrijndael)+'/
    crypto/fredmans21.txt.encrypted','/dev/null','d','
    9f9f90dbe3e5ee1218c86b8839db1995'],'
    mibenchrijndael_dec_fredmans21.txt'],
# 68
[mibenchrijndael,[os.path.dirname(mibenchrijndael)+'/
    crypto/track3.mp3.encrypted','/dev/null','d','9
    f9f90dbe3e5ee1218c86b8839db1995'],'
    mibenchrijndael_dec_track3.mp3'],
# 69
[mpeg2decode_cpp,['-q','-b',os.path.dirname(
    mpeg2decode_cpp)+'/input_base_4CIF_96bps.mpg','-o3',
    os.path.dirname(mpeg2decode_cpp)+'/output'],'
    mpeg2decode_cpp'],
# 70
[mpeg2decode,['-q','-b',os.path.dirname(mpeg2decode)+'/
    input_base_4CIF_96bps.mpg','-o3',os.path.dirname(
    mpeg2decode)+'/output'],'mpeg2decode'],
# 71
[python,[os.path.dirname(python)+'/../../sort.py'],'
```

```
        python_0'],
      # 72
      [python,[os.path.dirname(python)+'/../../primes.py'],'
        python_1'],
    ]
#   l += get_shootout_benches(top, res)
    '''todo:
      mpeg4dec.without_mmx
      mpeg4enc.without_mmx
      mpeg4dec.with_mmx
      mpeg4enc.with_mmx
    '''
    tasks = []
    if len(sys.argv) == 1:
      tasks = [ x for x in range(len(l)) ]
    else:
      if sys.argv[1] == 't':
        def hash_hack(h):
          for s in gPossibleRoots:
            h = h.replace(s,'ROOT')
          return h
        alreadyRunHashes = {}
        for x in m2s.cache:
          alreadyRunHashes[hash_hack(x)] = m2s.cache[x]
        for i in range(len(l)):
          now = l[i]
          bin=now[0]
          args=now[1]
          cpucache='cpucache_'+now[2]
          cpupipe='cpupipe_'+now[2]
          cmdhash = m2s.fix_run_args(bin,
                args,
                {'stdout':'/dev/stdout','stderr':'/dev/stderr
                    ',
                  'cpu-cache':cpucache,
                  'cpu-pipeline':cpupipe},
                [],'cpu.cfg')['cmdhash']
          #if i == -1:
          #  m2s.remove(cmdhash)
          cmdhash = getcmdhash(m2s,l[i])

          cmdhash = hash_hack(cmdhash)
          if cmdhash not in alreadyRunHashes:
            print('%d was not run'%(i))
          elif 0 != alreadyRunHashes[cmdhash]['return']:
            print('%d returned %d'%(i,alreadyRunHashes[
                cmdhash]['return']))
            print(cmdhash)
```

```
#         print(cmdhash)
#         if cmdhash not in m2s:
#           print('%d was not run'%(i))
#         elif 0 != m2s.cache[cmdhash]['return']:
#           print('%d returned %d'%(i,m2s.cache[cmdhash]['
    return']))
#           print(cmdhash)
      elif sys.argv[1] == 's':
        for i in sys.argv[2:]:
          i = int(i)
          data = m2s.cache[getcmdhash(m2s,l[i])]
          print(data['stderr'])
      elif sys.argv[1] == 'r':
        now = l[int(sys.argv[2])]
        bin=now[0]
        args=now[1]
        cpucache='cpucache_'+now[2]
        cpupipe='cpupipe_'+now[2]
        cmdhash = m2s.fix_run_args(bin,
              args,
              {'stdout':'/dev/stdout','stderr':'/dev/stderr',
                'cpu-cache':cpucache,
                'cpu-pipeline':cpupipe},
              [],'cpu.cfg')['cmdhash']
        m2s.remove(cmdhash)
      else:
        tasks = [ int(x) for x in sys.argv[1:] ]
    if tasks:
      l = [ l[x] for x in tasks]
      for now in l:
        bin=now[0]
        args=now[1]
        cpucache='cpucache_'+now[2]
        cpupipe='cpupipe_'+now[2]
        print(bin)
        print(args)
        m2s.run(bin,
                args,
                {'stdout':'/dev/stdout','stderr':'/dev/stderr
                  ',
                 'cpu-cache':cpucache,
                 'cpu-pipeline':cpupipe},
                [],'cpu.cfg')
main()

#!/usr/bin/env python
import re
from glob import glob
def main():
```

```
  cache = []
  pipeline = []
  general = []
  files = glob('*')
  for fname in files:
    if 'cpu_cache_report_' in fname:
      cache.append(fname)
    elif 'cpu_pipeline_report_' in fname:
      pipeline.append(fname)
    else:
      general.append(fname)
  d = {}
  for fname in general:
    s = open(fname).read()
    for w in ('Instructions','Memory','Cycles','
       BranchPredictionAccuracy'):
      if w not in d:
        d[w] = []
      print(s)
      v = re.search(w+' = (\\d+\\.?\\d*)',s).group(1)
      if w == 'BranchPredictionAccuracy':
        v = float(v)
      else:
        v = int(v)
      d[w].append(v)

  d['BranchPredictionAccuracy'] = [ d['Instructions'][i] *
     d['BranchPredictionAccuracy'][i] for i in range(len(d[
     'Instructions'])) ]
  d['Instructions'] = sum(d['Instructions'])
  d['BranchPredictionAccuracy'] = sum(d['
     BranchPredictionAccuracy']) / d['Instructions']
  d['Cycles'] = sum(d['Cycles'])
  d['Memory'] = max(d['Memory'])
  d['InstructionsPerCycle'] = d['Instructions'] / float(d['
     Cycles'])
  f = open('summary','wt')
  f.write('''Instructions = %s
Cycles = %s
InstructionsPerCycle = %s
BranchPredictionAccuracy = %s
Memory = %s
'''%(d['Instructions'],d['Cycles'],d['InstructionsPerCycle'
   ],d['BranchPredictionAccuracy'],d['Memory']))
  f.close()
main()
```

```
#!/usr/bin/env python2
from m2s import multi2sim
```

```
def main():
  m2s = multi2sim()
  for t in m2s.cache:
    if 0 != m2s.cache[t]['return']:
      print(t)
main()

#!/usr/bin/env python
def main():
  import m2s
  m2s = m2s.multi2sim()
  l = []
  for x in m2s.cache:
    #if m2s.cache[x]['return'] !=0 or ('gzip' in x and ('No
        such file or directory' in m2s.cache[x]['stderr']
      or 'not overwritten' in m2s.cache[x]['stderr'])):
    #if ('gzip' in x and ('No such file or directory' in
      m2s.cache[x]['stderr'] or 'not overwritten' in m2s.
      cache[x]['stderr'])) or ('usage: rijndael
      in_filename out_filename [d/e] key_in_hex' in m2s.
      cache[x]['stdout']) or ('xerces' in x and m2s.cache[
      x]['return']!=0) or ('FAILED to validate SAX' in m2s
      .cache[x]['stderr'] or 'FAILED to transform' in m2s.
      cache[x]['stderr']):
#   if m2s.cache[x]['return'] == 1 and 'validate_' in x:
    if filter(lambda y: y in x or y in m2s.cache[x]['stderr
      ']+m2s.cache[x]['stdout'], ('sort.py','primes.py')):
      print(x,m2s.cache[x]['return'])
      print(m2s.cache[x]['stderr'])
      l.append(x)
  if l:
    print('press␣enter␣to␣continue')
    raw_input()
    for x in l:
      m2s.remove(x)
main()

#!/usr/bin/env python
import os
import sys
import cPickle
from filelock import FileLock
from multiprocessing import Process
import subprocess
from glob import glob

class multi2sim:
  def __init__(self,scriptdir='.'):
    self.cachefile = os.path.join(scriptdir, 'm2scache.
```

```
      pickle')
    self.lockfile = os.path.join(scriptdir, 'm2s.lock')
    self.lock = FileLock(self.lockfile)
    self.cache = self.load()
    i = 0
    j = len(self.cache)
    for x in [x for x in self.cache]:
      i += 1
      if 'ROOT' not in x:
        print('updating %d/%d'%(i,j))
        y = self.cache[x]
        self.remove(x)
        for s in ['/home/luis/src/m2s','/mnt/lfgmillani/
            newer','/mnt/lfgmillani/new']:
          x = x.replace(s,'ROOT')
        self.cache[x] = y
  self.save()
  def load(self):
    try:
      self.lock.acquire()
      f = open(self.cachefile,'rb')
      cache = cPickle.load(f)
      f.close()
      self.lock.release()
    except IOError:
      cache = {}
    except KeyboardInterrupt:
      self.lock.release()
      raise KeyboardInterrupt
    return cache
  def save(self):
    try:
      self.lock.acquire()
      try:
        f = open(self.cachefile,'rb')
        filecache = cPickle.load(f)
        f.close()
      except IOError:
        filecache = {}
      if len(filecache) >= len(self.cache):
        for x in filecache:
          if x not in self.cache:
            self.cache[x] = filecache[x]
      f = open(self.cachefile,'wb')
      cPickle.dump(self.cache, f)
      f.close()
      self.lock.release()
    except KeyboardInterrupt:
```

```
      self.lock.release()
      self.save()
      raise KeyboardInterrupt
  def join(self, fname):
    modified = False
    self.lock.acquire()
    f = open(fname, 'rb')
    other = cPickle.load(f)
    f.close()
    for a in other:
      if a not in self.cache:
        self.cache[a] = other[a]
  modified = True
    if modified:
      self.save()
    self.lock.release()
  def remove(self, cmdhash):
    while 1:
      try:
        self.lock.acquire()
        try:
    f = open(self.cachefile,'rb')
    filecache = cPickle.load(f)
    f.close()
  except IOError:
    filecache = {}
  for x in filecache:
    self.cache[x] = filecache[x]
  self.cache.pop(cmdhash)
        f = open(self.cachefile,'wb')
  cPickle.dump(self.cache,f)
  f.close()
  self.lock.release()
  break
      except KeyboardInterrupt:
        pass
  def remove_failed(self):
    l = []
    for i in self.cache:
      if self.cache[i]['return'] != 0:
        l.append(i)
        print(i)
    try:
      self.lock.acquire()
      try:
        f = open(self.cachefile,'rb')
        filecache = cPickle.load(f)
        f.close()
```

```
      except IOError:
        filecache = {}
      if len(filecache) >= len(self.cache):
        for x in filecache:
          if x not in self.cache and x not in l:
            self.cache[x] = filecache[x]
      [ self.cache.pop(x) for x in l ]
      f = open(self.cachefile,'wb')
      cPickle.dump(self.cache, f)
      f.close()
      self.lock.release()
    except KeyboardInterrupt:
      self.lock.release()
      self.remove_failed()
      raise KeyboardInterrupt
  def fix_run_args(self, binary, binargs, output, simargs=
     None, cpuconfig=None):
    prevdir = os.getcwd()
    if cpuconfig!=None and cpuconfig[0]!='/':
      cpuconfig=os.path.join(prevdir,cpuconfig)
    for x in output:
      if output[x][0] != '/':
        output[x] = os.path.join(prevdir,output[x])
    newdir = os.path.dirname(binary)

    output = output.copy()
    if cpuconfig != None:
      cpuconfig = os.path.abspath(cpuconfig)
    for x in output:
      output[x] = os.path.abspath(output[x])
    if simargs == None or simargs == []:
      simargs = ['--cpu-sim', 'detailed']
      if cpuconfig != None:
        simargs.append('--cpu-config')
        simargs.append(cpuconfig)
      if 'cpu-cache' in output:
        simargs.append('--report-cpu-cache')
        simargs.append(output['cpu-cache'])
      if 'cpu-pipeline' in output:
        simargs.append('--report-cpu-pipeline')
        simargs.append(output['cpu-pipeline'])
    if '--report-cpu-cache' in simargs:
      output['cpu-cache'] = os.path.abspath(simargs[simargs
         .index('--report-cpu-cache')+1])
    if '--report-cpu-cache' in simargs:
      output['cpu-pipeline'] = os.path.abspath(simargs[
         simargs.index('--report-cpu-pipeline')+1])
    cmd = ['m2s'] + simargs + [ binary ] + binargs
```

```
    cmdhash = cmd[:]
    if '--report-cpu-cache' in cmdhash:
      cmdhash.pop(cmdhash.index('--report-cpu-cache')+1)
    if '--report-cpu-pipeline' in cmdhash:
      cmdhash.pop(cmdhash.index('--report-cpu-pipeline')+1)
    cmdhash = str(cmdhash)
    for s in ['/home/luis/src/m2s','/mnt/lfgmillani/newer',
       '/mnt/lfgmillani/new']:
      cmdhash = cmdhash.replace(s,'ROOT')
    return {'binary':binary,'binargs':binargs,'output':
       output,
            'simargs':simargs,'cpuconfig':cpuconfig,'
               prevdir':prevdir,'newdir':newdir,'cmd':cmd,'
               cmdhash':cmdhash}
  def run(self, binary, binargs, output, simargs=None,
     cpuconfig=None):
    '''
    prevdir = os.getcwd()
    if cpuconfig!=None and cpuconfig[0]!='/':
      cpuconfig=os.path.join(prevdir,cpuconfig)
    for x in output:
      if output[x][0] != '/':
        output[x] = os.path.join(prevdir,output[x])
    os.chdir(os.path.dirname(binary))

    output = output.copy()
    if cpuconfig != None:
      cpuconfig = os.path.abspath(cpuconfig)
    for x in output:
      output[x] = os.path.abspath(output[x])
    if simargs == None or simargs == []:
      simargs = ['--cpu-sim', 'detailed']
      if cpuconfig != None:
        simargs.append('--cpu-config')
        simargs.append(cpuconfig)
      if 'cpu-cache' in output:
        simargs.append('--report-cpu-cache')
        simargs.append(output['cpu-cache'])
      if 'cpu-pipeline' in output:
        simargs.append('--report-cpu-pipeline')
        simargs.append(output['cpu-pipeline'])
    if '--report-cpu-cache' in simargs:
      output['cpu-cache'] = os.path.abspath(simargs[simargs
         .index('--report-cpu-cache')+1])
    if '--report-cpu-cache' in simargs:
      output['cpu-pipeline'] = os.path.abspath(simargs[
         simargs.index('--report-cpu-pipeline')+1])
    cmd = ['m2s'] + simargs + [ binary ] + binargs
```

```
    cmdhash = cmd[:]
    if '--report-cpu-cache' in cmdhash:
      cmdhash.pop(cmdhash.index('--report-cpu-cache')+1)
    if '--report-cpu-pipeline' in cmdhash:
      cmdhash.pop(cmdhash.index('--report-cpu-pipeline')+1)
    cmdhash = str(cmdhash)
    '''
    newargs = self.fix_run_args(binary, binargs, output,
       simargs, cpuconfig)
    binary = newargs['binary']
    binargs = newargs['binargs']
    output = newargs['output']
    simargs = newargs['simargs']
    cpuconfig = newargs['cpuconfig']
    prevdir = newargs['prevdir']
    newdir = newargs['newdir']
    cmd = newargs['cmd']
    cmdhash = newargs['cmdhash']
    os.chdir(newdir)
    if cmdhash not in self.cache:
      print('working')
      result = {}
      print(cmd)
      p = subprocess.Popen(cmd,
          stdout=subprocess.PIPE,stderr=subprocess.PIPE)
      result['stderr'] = p.stderr.read()
      result['stdout'] = p.stdout.read()
      result['return'] = p.wait()
      print(cmd)
      for name in ('cpu-cache','cpu-pipeline'):
        if name in output:
          f = open(output[name], 'rt')
          result[name] = f.read()
          f.close()
      self.cache[cmdhash] = result
      self.save()
    if self.cache[cmdhash]['return'] != 0:
      print(self.cache[cmdhash])
      print('THIS IS BAD: "%s" returned non-zero value'%(
         cmd))
    for name in ('stdout', 'stderr', 'cpu-cache','cpu-
       pipeline'):
      if name in output:
        f = open(output[name], 'wt')
        f.write(self.cache[cmdhash][name])
        f.close()
    os.chdir(prevdir)
  def __contains__(self, x):
```

```
    return self.cache.__contains__(x)

#!/usr/bin/env python
import sys
import cPickle
def main():
  if len(sys.argv) < 3:
    print('usage: cachejoin src [src ...] dst')
    sys.exit(1)
  fin = {}
  for name in sys.argv[1:]:
    f = open(name, 'rb')
    d = cPickle.load(f)
    f.close()
    for x in d:
      if x not in fin:
        fin[x] = d[x]
  f = open(sys.argv[-1],'wb')
  cPickle.dump(f, fin)
  f.close()
main()
```

8.5 Módulo do Valgrind para Listar Métodos Executados

```
static
IRSB* ft_instrument ( VgCallbackClosure* closure,
                      IRSB* bb,
                      VexGuestLayout* layout,
                      VexGuestExtents* vge,
                      IRType gWordTy, IRType hWordTy )
{
  char * fnname[1024];
  char * filename[1024];
  char * dirname[1024];
  Bool dirnameAvailable;
  IRStmt* st;
  Addr a;
  int i;
  UInt line;
  for(i = 0; i < bb->stmts_used; ++i){
    st = bb->stmts[i];
    a = st->Ist.IMark.addr;
    line = -1;
    *fnname = *filename = *dirname = '\0';
    if(VG_(get_fnname_if_entry)(a, fnname, sizeof(fnname)))
      {
      VG_(get_fnname)(a, fnname, sizeof(fnname));
      VG_(get_filename_linenum)(a,
                                filename, sizeof(filename),
```

```
                                        dirname, sizeof(dirname),
                                        &dirnameAvailable, &line);
        VG_(printf)(
          "==%d==␣"
          "%s\t%s\t%s\t%d\t%8x\n",
          VG_(getpid)(), fnname, dirnameAvailable ? dirname :
              "???",
          filename, line, a
        );
      }
    }
    return bb;
}
```

REFERÊNCIAS

ABREU, F. Brito e; MELO, W. Evaluating the impact of object-oriented design on software quality. In: SOFTWARE METRICS SYMPOSIUM, 1996., PROCEEDINGS OF THE 3RD INTERNATIONAL. **Anais...** [S.l.: s.n.], 1996. p.90–99.

Advanced Encryption Standard (AES). Washington: National Institute of Standards and Technology, 2001. Federal Information Processing Standard 197.

BARKER, W. **Recommendation for the triple data encryption algorithm (TDEA) block cipher**. [S.l.]: US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2004.

BELLARE, M.; ROGAWAY, P. The exact security of digital signatures-How to sign with RSA and Rabin. In: ADVANCES IN CRYPTOLOGY—EUROCRYPT'96. **Anais...** [S.l.: s.n.], 1996. p.399–416.

BENJAMINI, Y.; HOCHBERG, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. **Journal of the Royal Statistical Society. Series B (Methodological)**, [S.l.], p.289–300, 1995.

CHIDAMBER, S.; KEMERER, C. A metrics suite for object oriented design. **Software Engineering, IEEE Transactions on**, [S.l.], v.20, n.6, p.476–493, 1994.

CORRÊA, U. Aplicação de métricas de software na predição de características físicas de software embarcado. , [S.l.], 2011.

EVERITT, B. **Cambridge Dictionary of Statistics . CUP**. [S.l.]: ISBN 0-521-81099-x, 2002.

GUTHAUS, M. et al. MiBench: a free, commercially representative embedded benchmark suite. In: WORKLOAD CHARACTERIZATION, 2001. WWC-4. 2001 IEEE INTERNATIONAL WORKSHOP ON. **Anais...** [S.l.: s.n.], 2001. p.3–14.

HENRY, S.; KAFURA, D. Software structure metrics based on information flow. **Software Engineering, IEEE Transactions on**, [S.l.], n.5, p.510–518, 1981.

INTEL. **Intel 64 and IA-32 Architectures Software Developer's manual**. [S.l.]: Intel, 2012. v.2.

KULKARNI, P. et al. Finding effective optimization phase sequences. In: ACM SIGPLAN NOTICES. **Anais...** [S.l.: s.n.], 2003. v.38, n.7, p.12–23.

LINCKE, R.; LUNDBERG, J.; LÖWE, W. Comparing software metrics tools. In: SOFTWARE TESTING AND ANALYSIS, 2008. **Proceedings...** [S.l.: s.n.], 2008. p.131–142.

MARTIN, M. et al. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. **ACM SIGARCH Computer Architecture News**, [S.l.], v.33, n.4, p.92–99, 2005.

MCCABE, T. A complexity measure. **Software Engineering, IEEE Transactions on**, [S.l.], n.4, p.308–320, 1976.

REINER, A.; YEKUTIELI, D.; BENJAMINI, Y. Identifying differentially expressed genes using false discovery rate controlling procedures. **Bioinformatics**, [S.l.], v.19, n.3, p.368–375, 2003.

Secure Hash Standard. Washington: National Institute of Standards and Technology, 2012. Federal Information Processing Standard 180-4.

SIMON, F.; STEINBRUCKNER, F.; LEWERENTZ, C. Metrics based refactoring. In: SOFTWARE MAINTENANCE AND REENGINEERING, 2001. FIFTH EUROPEAN CONFERENCE ON. **Anais...** [S.l.: s.n.], 2001. p.30–38.

SIMPSON, M.; MIDDHA, B.; BARUA, R. Segment protection for embedded systems using run-time checks. In: COMPILERS, ARCHITECTURES AND SYNTHESIS FOR EMBEDDED SYSTEMS, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.66–77.

STIGLER, S. **The history of statistics**: the measurement of uncertainty before 1900. [S.l.]: Belknap Press, 1986.

UBAL, R. et al. Multi2Sim: a simulation framework to evaluate multicore-multithread processors. , [S.l.], 2007.

UBAL, R. et al. **The Multi2Sim Simulation Framework**. 2011.

UBAL, R. et al. Multi2Sim A Simulation Framework for CPU-GPU Computing . In: INTERNATIONAL CONFERENCE ON PARALLEL ARCHITECTURES AND COMPILATION TECHNIQUES, 21. **Proceedings...** [S.l.: s.n.], 2012.

WOLBERG, J.; WOLBERG, J. **Data analysis using the method of least squares**: extracting the most information from experiments. [S.l.]: Springer Berlin, Germany, 2006. v.1.