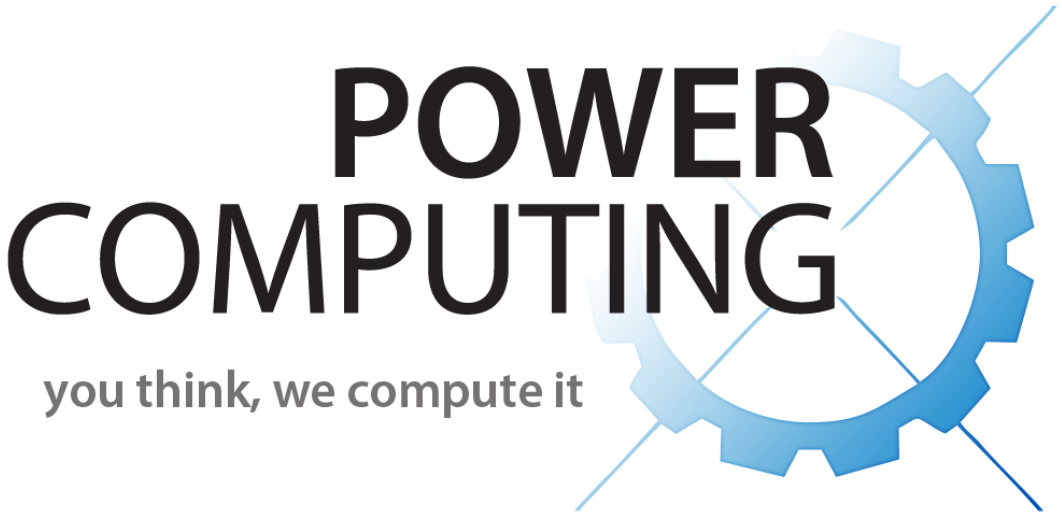
Relatório Final



Projecto:	**Life Inspiration**
Autor:	Power Computing
Data de preparação:	2012-06-07
Circulação:	Power Computing – power.computing.psi@gmail.com

Índice

1 Elementos da empresa

11127 - Bruno Oliveira

13678 - Filipe Cardoso

15185 - Francisco Nunes

11161 - João Veríssimo

15555 - Miguel Miranda

11340 - Paulo Lemos

15506 - Pedro Leal

15510 - Rodrigo Marinheiro

13896 - Tiago Duarte

2 Objectivos

2.1 Tarefas atribuídas á power Computing

O objectivo da power computing neste projecto é gerir os recursos informáticos de modo a calcular de modo intensivo, algoritmos genéticos, mostrando em tempo real ao utilizador o estado do processo. Esta gestão engloba a criação de uma comunicação com a Optimum Computing e calcular os problemas recorrendo aos algoritmos desenvolvidos por a Inovation e Genetic Lab.

Os pontos que devem ser implementados por a Power Computing são:

- O processamento deverá ser paralelo, distribuído e balanceado pelas máquinas disponíveis.

- Capacidade de integrar os métodos que venham do departamento Genetic Lab, no sistema, de forma dinâmica e abstracta.

- Capacidade de receber o pedido de execução através de uma interface web criada por a Optimum Computing, com toda a parametrização necessária para correr o problema.

- Capacidade de manter informada a interface web da Optimum Computing, do estado actual do problema utilizando a tecnologia NodeJS para comunicação de notificações em tempo real, e uma base de dados para armazenamento, calculo estatístico e futura análise do problema. e do estado dos servidores e a sua disponibilidade.

2.2 Grau de cumprimento dos objectivos.

O grupo da Power Computing conseguiu concluir praticamente todos os desafios que nos foram propostos, sendo que, a única limitação foi a não implementação de distribuição e balanceamento devido a não ser possível concluir em tempo útil uma versão estável de todo o software.

3 Técnica do trabalho desenvolvido.

3.1 Reflection

Para facilitar a implementação dos algoritmos implementados pela Genetic Lab foi criado uma estrutura que utiliza a tecnologia do java Reflection. Esta estrutura é utilizada para carregar todos os algoritmos e executa-los de forma dinâmica. Deste modo é possível criar uma abstracção dos problemas, uma vez que para executar o código criado pelo grupo da Genetic Lab apenas é necessário substituir o .jar (conjunto de ficheiro classes) no servidor da Power Computing.

O Reflection é utilizado para ler toda a informação de Problemas e Operadores existentes no código que por sua vez é passada para o Optimum Computing através do módulo de Node.JS.

Outro modo de utilização do Reflections na estrutura do Power Master é a inicialização dos solvers da Genetic Lab, esta inicialização está implementada de forma a que não seja necessário criar mais código sempre que existe um novo problema ou operador disponível no código da Genetic Lab.

Sempre que existe um novo operador ou problema apenas é preciso passar informação lida com o Reflection para a Optimum Computing e de seguida receber pedidos através da interface Web utilizando o WorkSocket, e inicializar os solvers com os parâmetros introduzidos nessa interface Web.

A passagem de parâmetros é passada directamente para os solvers utilizando o Reflection, não existindo qualquer intercepção de informação excepto o parâmetro de mutação que exige alteração dinâmica do mesmo.

3.1.1 Tabelas:

- TaskLoader

TaskLoader
ClassName: String **ClassObject: Class**
TaskLoader () **getClassObject : Class** **FileToByteArray: byte []** **CreateClass: Class**

- GeneticLoader

GeneticLoader
jarFile : String **genericList : Map** **jcl : org.xeustechnologies.jcl.JarClassLoader**
GeneticLoader() **loadClasses : void** **getConstructors: String** **getInfo: String** **loadClasses: ArrayList** **getAlgorithms : ArrayList** **getInfoJSON: String** **getSolver : GenericSolver** **getInfo: ArrayList**

- Base64Coder

Base64Coder
systemLineSeparator : java.lang.String **map1 : char[]** **map2 : byte[]**
Base64Coder() **encodeString: String** **encodeLines: String** **encode: char []** **decodeString: String** **decodeLines: byte []** **decode: byte []**

3.2 NodeJS e Calculo Assíncrono

Para ser possível mostrar em tempo real o decorrer dos problemas, foi necessário implementar a tecnologia do NodeJS na interface Web e como consequência na estrutura do Power Master. Assim sendo foi implementado através de uma biblioteca com nome de SocketIO um módulo que permite o envio em tempo real para a página Web da Optimum Computing, que esta representado na classe NodeEmiter. Este modo de comunicação é utilizando durante a computação dos solvers, para assim ser enviada a informação sobre o estado dos solvers assim como o envio de todos os cálculos estatísticos e resultados finais do solvers.

O cálculo assíncrono é um modo de obter informação dos diversos solvers em execução sem que exista qualquer método de sincronização. Este módulo de cálculo esta implementado na classe AsyncSats. Este objecto é responsável por criar os diversos solvers, retirar informação dos solvers, inserção de dados na base de dados já calculados, paragem de solvers e actualização de parâmetros no solver.

3.2.1 Tabelas:

- AsyncStats

AsyncStats
numThreads : java.util.concurrent.atomic.AtomicInteger
-period : int
-aux : int
-idClient : int
-idProblem : int
-Stop : boolean
-arrayThread : powermaster.SolverThread[]
-BestToFound : int
-controlo : int
AsyncStats
Stop() : void
getBestPopulation() : String
UpdateParametrs: void
getAllUniqueIndividuals: String
getAllUniqueIndividualsCount: int
getBestIndividual() : double
run() : void

- NodeEmiter

NodeEmiter
socket : io.socket.SocketIO
NodeEmiter()
Emit: void
EmitStop: void
EmitPop: void
EmitInfo: void
EmitStatus: void
ReconectSingle: void
Reconect : void
onDisconnect : void
onConnect : void
onMessage:void
on : void
onError: void

3.3 Powermaster

Aplicação que permite correr os algoritmos criados por a Genetic Lab, assim como responder aos vários pedidos da Optimum Computing.

O objecto SaveStatus serve para guardar um conjunto de populações de diversos solvers, sendo que é útil para a leitura de informação sem recorrer à base de dados.

O objecto GeneticEvents serve para os algoritmos da Genetic Lab conseguirem comunicarem com a estrutura da Power Computing.

O GeneticEvents tem uma estrutura com eventos de modo a que sempre que um solver inicia e, durante a sua execução invoca métodos que informa as classes da Power Computing acerca do estado dos solvers.

Dentro desses eventos são efectuados os cálculos estatísticos e inserções na base de dados.

O objecto SolverThread é onde são executados os solvers da Genetic Lab. Cada objecto SolverThread é composto por uma ligação à base de dados dedicada para realizar a inserção assim como um objecto próprio pertencente à GeneticEvents.

3.3.1 Tabelas:

- SaveStatus

SaveStatus
Name: String **pops: ArrayList** **tipo: int**
SaveStatus **AddPopulation: void** **getNumPopulations: int** **getPopulation: Population** **getBestPopulation: String**

- PowerMaster

PowerMaster
arrayThread : SolverThread[] NUM_THREADS: int INTERVAL_PART: int
PowerMaster() main: void

- GeneticEvents

GeneticEvents
Interval : int nextInterval : int idClient : int idProblem : int db: Database
GeneticEvents EventStartSolver: void EventIteraction: void EventFinishSolver: void

- SolverThread

SolverThread
solver : GenericSolver numThreads : AtomicInteger
SolverThread getSolver() : genetics.GenericSolver Stop: genetics.Population getPopulation: genetics.Population getUniqueIndividuals : java.util.Collection run : void

3.4 Module

Para facilitar o desenvolvimento, foi criado uma estrutura básica que permite fazer referenciação de objectos por toda a estrutura e com detecção de erros na inicialização do software. Cada adição na estrutura tem o nome de módulos sendo que existe os seguintes módulos:

- Base de dados;
- Administração;
- WebHTTP (Comunicação Optimum Computing)
- Node.JS
- Estatística;

3.4.1 Tabelas:

- GlobalData

GlobalData
database_user : java.lang.String **database_pass : java.lang.String** **database_location : java.lang.String** **database_database : java.lang.String**
GlobalData()

- AbstractAplication

AbstractAplication
AplicationStatus : boolean **AplicationName : java.lang.String**
AbstractAplication **getAplicationName : java.lang.String** **getAplicationStatus : boolean**

- Aplication

Aplication
STATUS: boolean **db : Database** **admin : Administration** **workSocket : WorkSocket** **nodeJS : NodeEmiter** **AplicationStatus: Hashtable**
Aplication **iniModules : boolean** **getApplicationSatus : boolean**

3.5 Web http

De forma a receber pedidos da interface Web, foi criado um módulo que contém um socket no porto 8080 que recebe todos os problemas e pedidos especiais. Neste porto é feito a gestão do pedido, onde de seguida é efectuada a resposta através do módulo NodeJS. O módulo WebHTTP tem um sistema multi-thread para suportar vários pedidos em simultâneo. Sempre que existe um novo pedido é criado uma thread com o objecto newClient onde é processado esse pedido. Caso seja um pedido novo de um problema é criado um objecto assíncrono que irá iniciar os vários solvers e começar a o seu processamento. O objecto assíncrono para criar os diversos solvers vai usar o objecto SolverCreator para interpretar os parâmetros passados pela interface Web e depois os passar para os solvers. O objecto SolverCreator também é responsável por alterar o parâmetro de mutação dinamicamente.

3.5.1 Tabelas:

- SolverCreator

SolverCreator
SolverCreator CreateSolver: GenericSolver

- WorkSocket

WorkSocket
port : int ss : ServerSocket clients : Map
WorkSocket run: void getClientsData: String client: java.util.Map

- newClient

newClient
Cliente: Socket async : AsyncStats idClient : int id : int
newClient StopSolver: void getPopulation : void UpdateParms: void getBestPopulation : void run : void

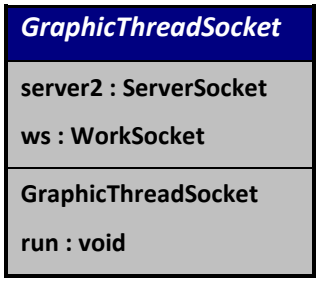
3.6 Administration

O módulo de administração serve para o gestor que está monitorizar os servidores conseguir retirar informação mais rapidamente e sem grandes complexidades. Este módulo abre dois sockets nas portas 666 e 667 para que outra aplicação com o nome de Admin Tool consiga ligar-se e obter informações e controlar o servidor.

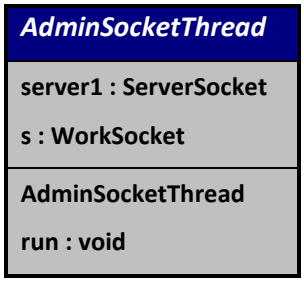
As informações que são possíveis retirar são, o uso da memória, relativas ao CPU, assim como problemas em execução. Com esta ferramenta também é possível enviar testes para o servidor assim como iniciar o servidor, e executar comandos na linha de comandos do sistema operativo.

3.6.1 Tabelas:

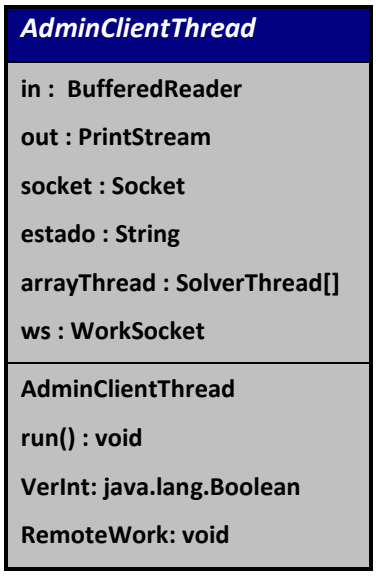
- GraphicThreadSocket



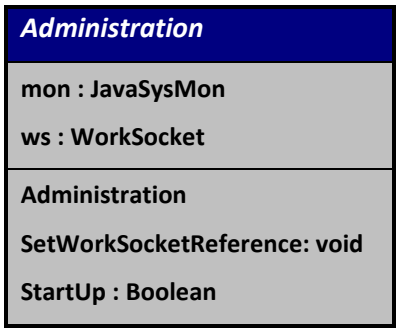
- AdminSocketThread



- AdminClientThread



- Administration

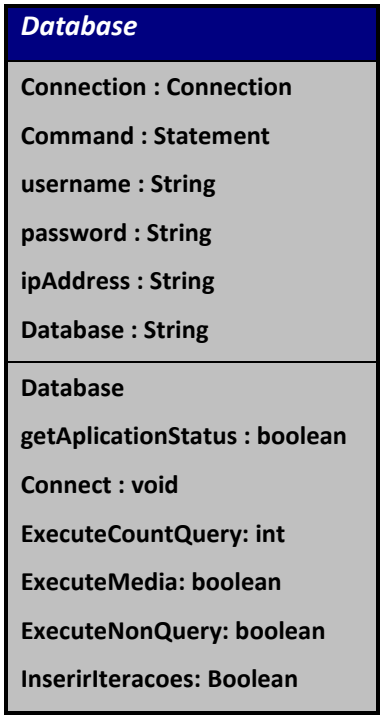


3.7 DataBase

O módulo de bases dados é onde estão implementadas todas as operações de inserção e leitura efectuadas sobre a base de dados. Este módulo tem como função fazer as ligações com a base de dados e executar querys aos servidores de mysql.

3.7.1 Tabelas:

- Database



4 Ferramentas proprietárias

4.1 ReflectionGUI

O ReflectionGUI é uma aplicação em java que utiliza java reflection para carregar de forma dinâmica os algortimos da genetic lab, sendo assim esta ferramenta permite à genetic lab fazer testes para demonstrar que os solvers não contem erros, de forma rápida e fácil, uma vez que para executar o código criado pelo grupo da Genetic Lab, apenas é necessário substituir o .jar existente na pasta da aplicação.

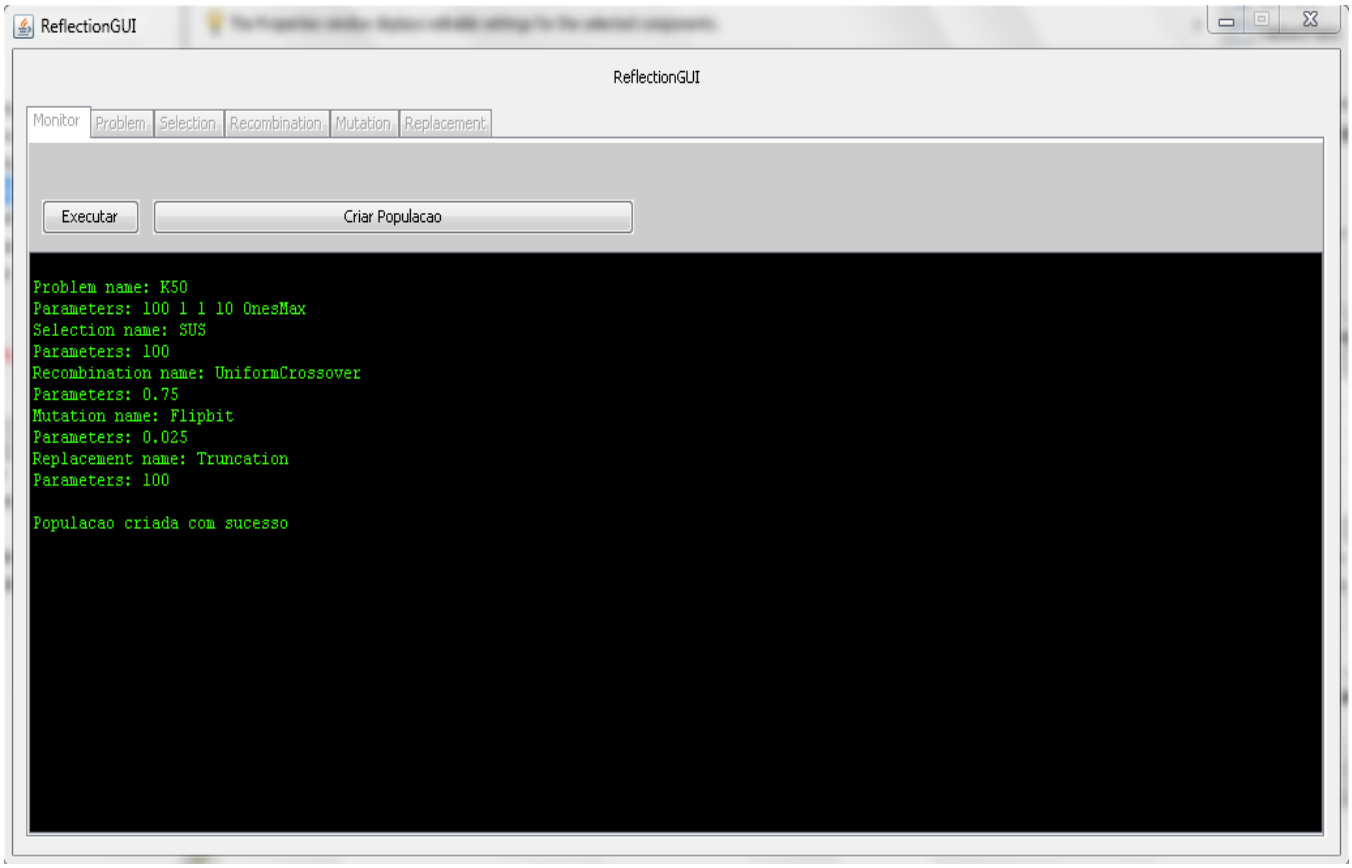


Imagem 1 – Janela Reflection GUI

4.2 Sender

O Sender é uma aplicação simples que permite fazer diversos testes na power computing, durante o desenvolvimento do trabalho, pois permite enviar trabalho simulando um cliente de forma fácil e rápida, permitindo assim, detectar erros existentes no código.

Imagem 2 – Janela Sender GUI

4.3 Admim Tool

A admin tool permite a todos os utilizadores que possuem conta na mesma monitorizar o servidor da powercomputing.

De todas as funcionalidades da admin tool destacam-se as seguintes:

- Lançamento de pequenas demos dos problemas para testar os algoritmos;
- Monitorização do desempenho do servidor a nível de RAM e CPU;
- Ligar o servidor;
- Executar comandos no servidor.

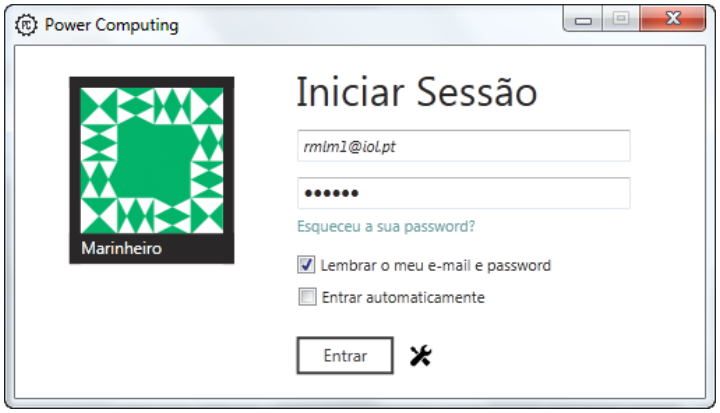


Imagem 3 – Janela de autenticação na admin tool

Para utilizar a admin tools é necessário realizar a autenticação na aplicação pois existem funcionalidades que devem ser protegidas, como por exemplo a paragem de execução de problemas.

As demos são executadas através de um pedido para o porto 8080 do servidor da Power Computing, tal como a Optimum faz com os pedidos dos clientes, permitindo-nos assim ter acesso a todos os resultados de forma rápida.

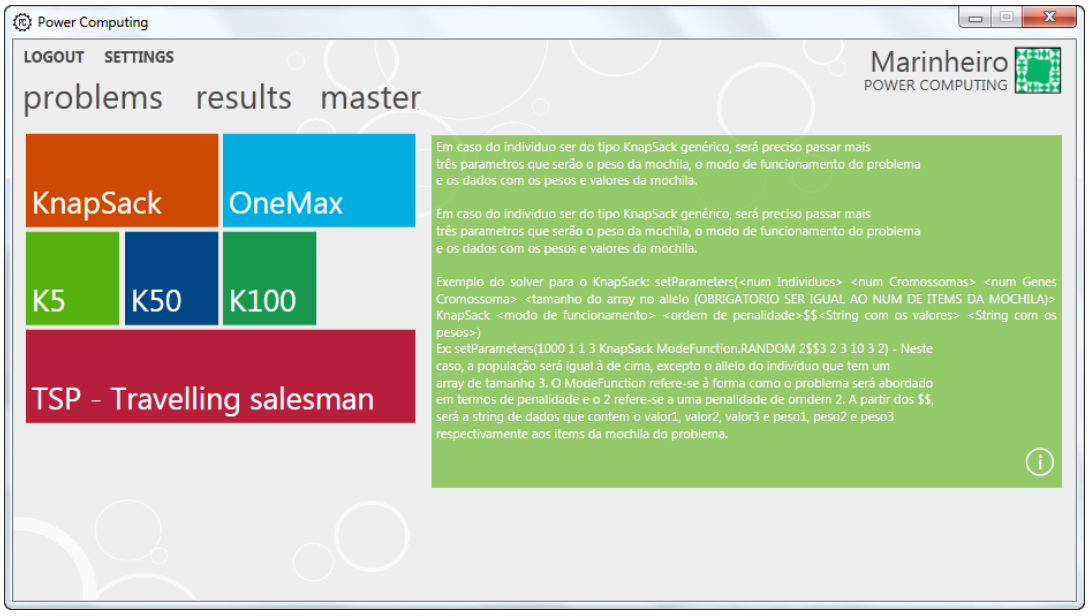


Imagem 4 – Área de lançamento das demos

Já no que diz respeito á monitorização do desempenho a admin tool faz pedidos através de um socket para o porto 667 do servidor da powercomputing, que através do seu módulo de administração recolhe os dados que o script feito para tal tarefa gera (CPU), conseguindo assim construir um gráfico com informação acerca do consumo de RAM e CPU.

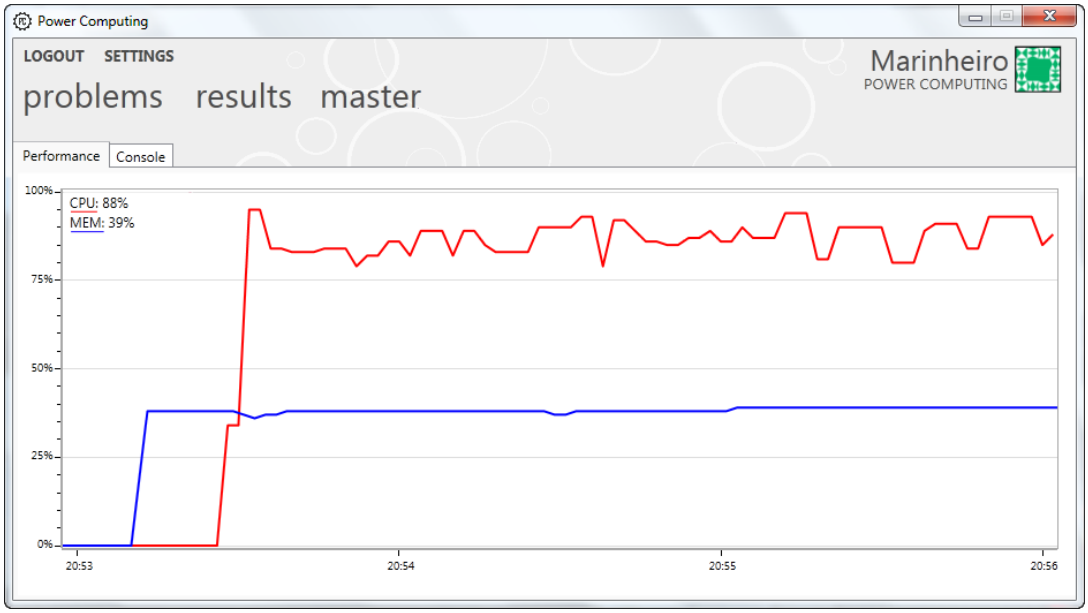


Imagem 5 – Gráfico do desempenho do servidor

A Admin Tool permite ainda a execução de alguns comandos essenciais á monitorização do sistema.

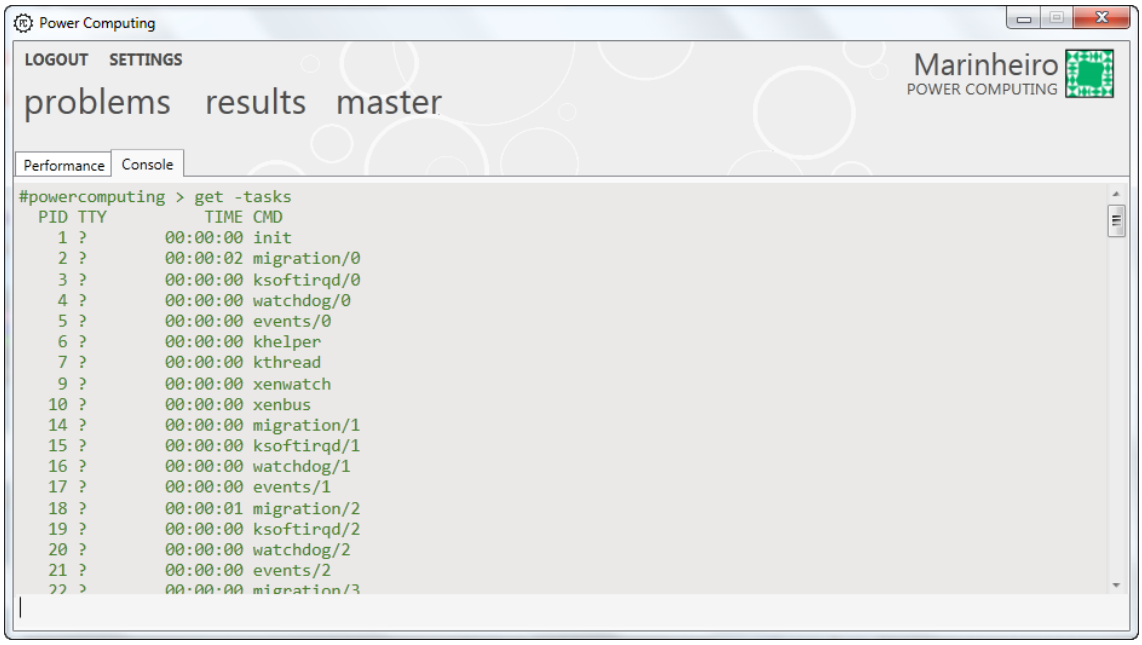


Imagem 6 – Consola para execução de comandos no servidor

4.4 Web site

4.4.1 Objectivos

- Disponibilizar atempadamente a todos os elementos do departamento todos os documentos importantes referentes á empresa;

- Dar a conhecer quais as principais funcções do nosso departamento bem como os elementos que nele trabalham;

- Permitir que o utilizador possa ligar o serviço PowerMaster.jar sem ter de aceder directamente ao servidor, bem como saber se este já está a correr e quais os processos que estão a correr de momento.

4.4.2 Material utilizado

- Recorreu-se ás linguagens php, html e java;
- Foi utilizado cgi.

4.4.3 Procedimentos de utilização

A utilização do website é extramamente simples. Na página inicial encontra-se uma breve descrição acerca dos objectivos do departamento bem como uma breve descrição acerca do projecto em si. Para além disso encontram-se ainda descritos todos os elementos integrantes do departamento bem como a descrição de quais as suas funções. Na mesma página encontram-se disponíveis avisos de importância aos utilizadores e encontram-se ainda documentos disponíveis para os elementos do departamento poderem consultar e descarregar.

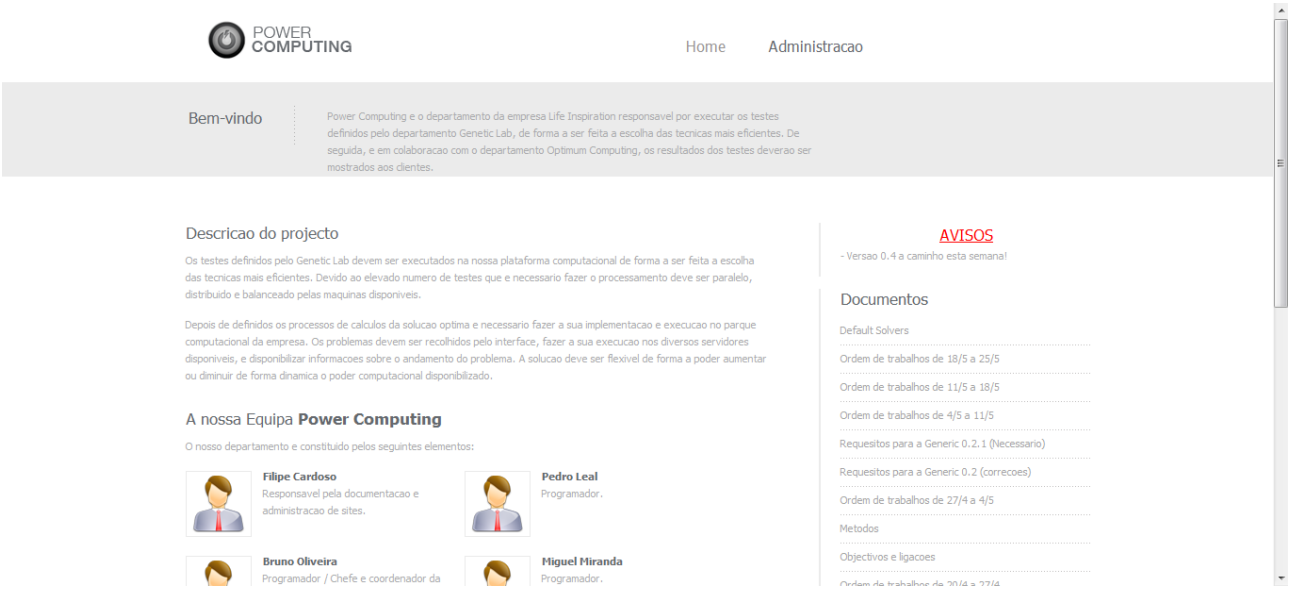


Imagem 7 – Pagina inicial da Website

Na página de Administração podemos visualizar de imediato duas áreas, cada uma com descrição acerca de qual a sua função, onde é possível ao utilizador, ao clicar num dos botões presentes na página, descarregar uma pequena aplicação em java que lhe irá permitir monitorizar quais os serviços que estão a ser executados no servidor ou então arrancar o ficheiro PowerMaster.jar. Visto até à data não ter sido disponibilizado qualquer porta para poder efectuar a comunicação entre cliente e servidor, embora já tenho sido solicitado, não foi possível executar testes a fim de verificar o funcionamento destas aplicações, pelo que não está garantido o funcionamento integral das mesmas.

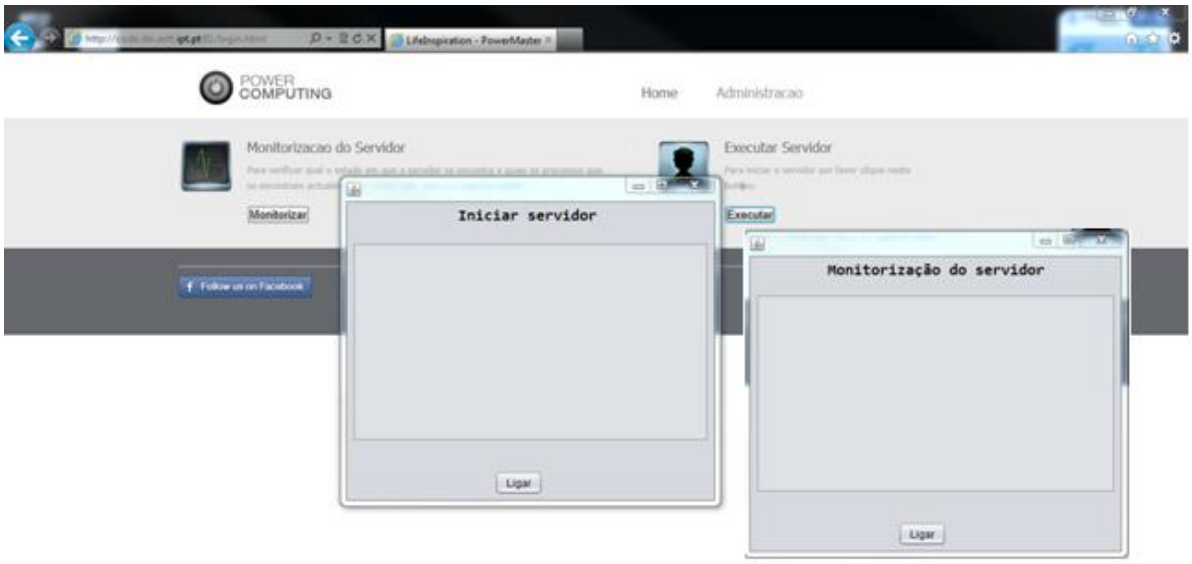


Imagem 8 – Pagina de Administração da Website

A ideia inicial seria a utilização de CGI para executar scripts do lado do servidor e o seu resultado fosse mostrado no browser ao utilizador. Tal não foi possível de ser efectuado pois embora o CGI estivesse instalado no servidor não corria, problema esse que não foi possível resolver;

O próximo passo seria a utilização da linguagem php para tentar resolver o problema dos scipts, e até mesmo poder ser implementado um pequeno fórum para que os membros do departamento pudessem rapidamente trocar informação entre si. Também não foi possível colocar a linguagem php a correr no servidor;

Por fim foram elaboradas 3 aplicações, 2 para o lado do cliente e outra para o lado do servidor, de modo a remediar de forma rápida os problemas e atrasos encontrados, tendo este último ponto sido efectuado parcialmente, pois como já foi referido anteriormente não foi possível efectuar testes pois não foi atribuído qualquer porta para a aplicação do lado do servidor.

5 Descrição das tarefas

5.1 O que vamos enviar:

- Enviaremos através do NodeJS avisos para a Optimum Computing, quando são calculados e guardados resultados novos na base dados.

5.2 O que vamos receber:

- Iremos receber uma string normalizada segundo as características definidas nos requisitos e formatada através o standard do JSON, com todos os parâmetros necessários para resolver o problema.
- Após a recepção o sistema inicia automaticamente todos os Solvers.

5.3 O que vamos guardar:

Base de dados:

Uma vez que vários solvers irão correr em simultâneo é necessário guardar alguns dados como por exemplo o desvio padrão e a variância assim como o valor do melhor individuo e seus atributos em cada iteração efectuada.

Enviar todos estes dados em bruto para a interface seria impraticável, portanto é então necessário tirar algum valor e relevância desses dados. Através do método assíncrono criado para este propósito, são efectuadas médias e outros cálculos necessários para que a optimum computing possa mostrar dados do progresso dos solvers na sua interface.

Como?

Cada thread irá correr um solver que fará várias iterações. Uma thread corresponde a uma população. Toda a informação de cada iteração será guardada numa base de dados com informação sobre todos os parâmetros de interesse nela:

- Identidade do Cliente;
- Identidade do Problema;
- Tempo em que foi executada (TimeStamp);
- Valor do melhor indivíduo da iteração;
- Média dos indivíduos;
- Atributos do melhor indivíduo da iteração e valor do melhor indivíduo;
- Desvio Padrão da população;
- Tipo de iteração (se 0 é uma thread na sua primeira iteração, se 1 é uma thread a trabalhar, se 2 é uma thread na sua ultima iteração);
- Na última iteração é guardada uma string com todos os indivíduos únicos em todos os solvers.
- Variância;

No fim de um número pré-determinado de iterações, serão escrito na base de dados de resultados, os valores globais de todas as threads:

- Identidade do Cliente;
- Identidade do Problema;
- Média de todas as populações;
- Média do desvio de todas as populações;
- Melhor de todas as populações;
- Número de melhores fitness únicos de todas as populações;
- Variância média de todas as populações;
- Resultado final enviado na ultima iteração;

Quando o resultado é inserido na tabela dos resultados, é enviado um aviso através do nodeJS a dizer que se encontram disponíveis novos resultados. Após ser recebido esse aviso, a Optimum Computing vai fazer query na base de dados para mostrar os resultados na sua interface.

5.3.1 Estrutura das tabelas

Atributos da tblIterations

threadId: int – Thread correspondente a iteração;
itera: int – Número da iteração;
idClient: int – Identificador do cliente;
idProblem: int – Identificador do problema;
time: datetime – Timestamp da inserção;
best: double – Fitness do melhor individuo;
average: double - Média de fitness da população;
atributos: longtext – Atributos do melhor individuo
deviation: double – Desvio padrão da população
type: tinyInt – Tipo de iteração (0-Primeira iteração;1-Iterações subsequentes;2-Iteração final)
variance: double – Variância

Atributos da tblResults

itera: int – Número da iteração;
idClient: int – Identificador do cliente;
idProblem: int – Identificador do problema;
globalAverage: double – Média global de todas as threads;
globalDeviation: double – Desvio padrão de todas as threads;
globalBest: double – Melhor individuo naquele momento;
globalNumBest: int – Número de indivíduos iguais ao melhor;
variance: int – Variância de todas as threads;
final longtext – String final com os atributos de todos os indivíduos únicos de todos os solvers.

Atributos da logins

idLogin: int – ID único que identifica o utilizador;
name: varchar – Nome do utilizador;
email: varchar – Email do utilizador;
password: varchar – Password encriptada em SHA1 do utilizador;
salt: varchar – salta para reforçar a segurança da encriptação da password;
premission: int – Nível de permissões do utilizador;
active: int – Flag que identifica se a conta está activa ou não;
date: timestamp – Data e hora em que a conta foi criada.

Modelo de entidade e relacionamento

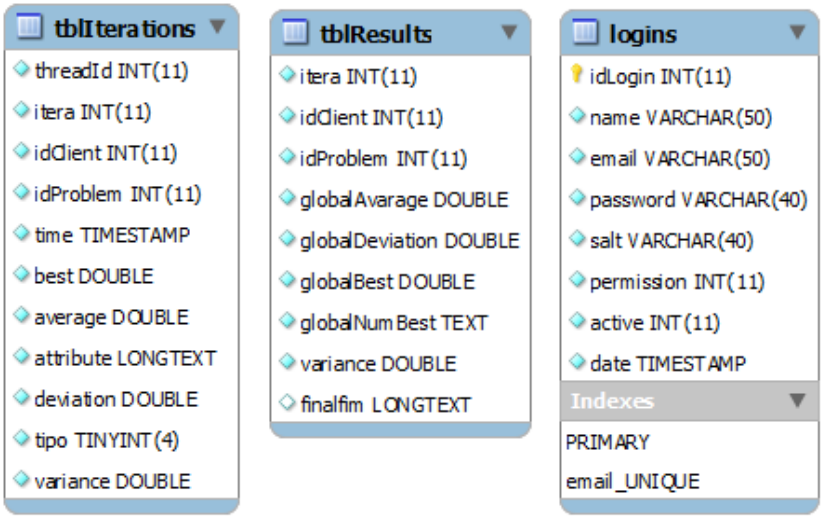


Imagem 8 – Tabelas da base de dados

6 Conclusão

Apos a conclusão deste projecto, adquirimos vários tipos de experiencias. Experiencia em trabalhar entre varias equipas, e a fazer parte de uma projecto onde todas dependiam do trabalho individual para alcançar o objectivo comum, algo que por vezes era difícil devido à comunicação tanto internamente como com os outros colaboradores. Experiencia em organizar as tarefas pelos membros da equipa o que nem sempre era fácil e experiencia em lidar com tecnologias novas, como o Java Reflecion que foi usado para carregar todos os algoritmos e executa-los de forma dinâmica. Aprendemos também a trabalhar com node.js com a finalidade de mostrar em tempo real o decorrer dos problemas, e para guardar a informação preciosa acerca de cada problema foi usado MySQL.

Esta experiência pode vir a ser útil no futuro, pois além dos conhecimentos sobre outras tecnologias, tivemos uma amostra de uma ambiente de trabalho numa empresa, com prazos, exigências todas as semanas e colegas de trabalho, por isso é positivo ter passado por este projecto para estar preparado para tal.

7 Contactos

Power Computing – power.computing.psi@gmail.com

Ou

www.facebook.com/PowerComputing