UNIVERSIDADE DE COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA ***Departamento de Engenharia Informática***	**Trabalho Prático N.º 2** **Computação de Alto Desempenho** **2012/13 – 2º Semestre** MEI **Deadline: 3-6-2013**

<u>Nota</u>: A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante do ensino superior e futuro profissional. Qualquer tentativa de fraude pode levar à reprovação na disciplina tanto do facilitador como do prevaricador.

Message Passing Interface – Parallel Programming for Clusters

Project Goals

Learn the Message Passing Interface for High Performance Computing.

Description of the Project

In the second project, students should use the Message Passing Interface (MPI) or some other parallelization technology for clusters, to solve the same "Grand Rule Challenge". The use of MPI may force some changes to the first project, starting from the programming language, which must now be C. The most important points for the evaluation of this work are the quality of the MPI solutions and the speedup they achieve compared with a base single machine implementation. The report should include an evaluation of performance against the number of processes running. Students should evaluate speedups and efficiency of their solutions.

Students may use the following machines for their work:

```
cad-server01.dei.uc.pt
cad-server02.dei.uc.pt
cad-server03.dei.uc.pt
cad-server04.dei.uc.pt
cad-server05.dei.uc.pt
cad-server06.dei.uc.pt
cad-server07.dei.uc.pt
cad-server08.dei.uc.pt
cad-server09.dei.uc.pt
```

They may login using the student.dei.uc.pt credentials. Each of these machines has the entire dataset mounted in the directory `/dataset`. Students may take advantage of this fact to parallelize access to the files. As in the previous assignment, students should **optimize execution for a single transactions file**. The program should receive the rules file, the transactions file and the output file as command line parameters, **in this order**.

To avoid contention in the use of these machines students are advised to develop in their own machines as much as possible and reserve the common machines for testing and benchmarking only.

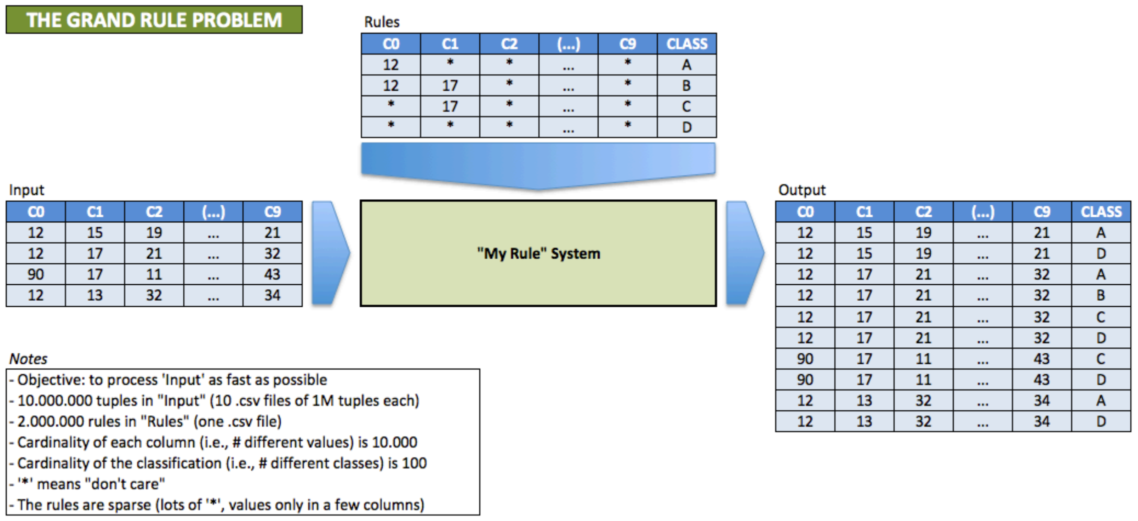
The Grand Rule Problem (Again)

Note: this problem was proposed and written by Professor Paulo Marques.

For self-containment, we repeat the description of the "Grand Rule Problem".

An industrial client (a bank) has recently approached a company in our town to classify the expenses of its clients according to a number of rules. In practice, for each bank transaction of an input file, they want to see if it matches a number of predefined rules. A bank transaction is a tuple with 10 attributes – *(a_0, a_1, a_2, a_3, …, a_9)*, where each attribute is an integer (32 bit). Each rule is also a tuple with 10 attributes plus a classification "c". I.e., *(r_0, r_1, r_2, r_3, …, r_9,c)*. Each rule attribute can either be a number (32 bit), or a "*", which means "don't care". For example, the transaction (1,2,3,4,5,6,7,8,9,0) matches the rule (1,2,3,4,5,6,7,8,9,0,111) and the rule (1,*,*,*,*,*,*,*,*,0,222), but it does not match either the rules (2,2,3,4,5,6,7,8,9,0,333) or (2,*,*,*,*,*,*,*,*,0,444).

Your challenge is to implement a system that allows efficiently solving this problem for large amounts of data. The next image illustrates the problem.



Rules

C0	C1	C2	(...)	C9	CLASS
12	*	*	...	*	A
12	17	*	...	*	B
*	17	*	...	*	C
*	*	*	...	*	D

Input

C0	C1	C2	(...)	C9
12	15	19	...	21
12	17	21	...	32
90	17	11	...	43
12	13	32	...	34

Output

C0	C1	C2	(...)	C9	CLASS
12	15	19	...	21	A
12	15	19	...	21	D
12	17	21	...	32	A
12	17	21	...	32	B
12	17	21	...	32	C
12	17	21	...	32	D
90	17	11	...	43	C
90	17	11	...	43	D
12	13	32	...	34	A
12	13	32	...	34	D

Notes
- Objective: to process 'Input' as fast as possible
- 10.000.000 tuples in "Input" (10 .csv files of 1M tuples each)
- 2.000.000 rules in "Rules" (one .csv file)
- Cardinality of each column (i.e., # different values) is 10.000
- Cardinality of the classification (i.e., # different classes) is 100
- '*' means "don't care"
- The rules are sparse (lots of '*', values only in a few columns)

In particular, the objective is to process 10 input files containing the transactions of the last 10 days. Each input file has 1M tuples. The rule table is fixed and has 2M rules. Most of the rules are actually empty (i.e., with "*"), having numbers in only a few columns of each rule. The number of unique values in each column is 10.000 and the number of different classifications is 100. Your solution should process the data as efficiently as possible, generating an output file for each input one. By "efficiently" we mean:

- Having a reasonably good algorithm for performing the computation.
- Utilizing the resources of a machine to their fullest.
- If you use a distributed approach, even so, use each machine to its full potential.

Evaluation Criteria

The evaluation of the work will be based on the following criteria:

- Correction & Clarity of the code. Students should ensure that their algorithm is working properly. Additionally, students should keep their code as clean as possible, despite using MPI.
- Performance improvement. Students should maximize the speedup of their MPI-based parallel version.
- Different proposals using MPI primitives. Evaluation will benefit works with multiple MPI communication solutions. It is however very important to document the results of these solutions in terms of performance. For this, students should include plots of speedup against the number of processes.

Report of the Work

Besides the software, students should also write a report describing their work. It should contain the following sections:

(A) Introduction.
(B) Description of the base sequential solution.
(C) Description of the parallelization approaches.
(D) Analysis of performance. This should include plots of speedup against the number of processes.
(E) Conclusions.

The report should be made as small as possible, yet with enough length to be clear and readable.

Deadline

Deadline to deliver the assignment: see the header.

Assignment upload

Students should deliver everything, including the report, in a .zip file. This file should include a README file with all the INFORMATION NECESSARY to execute and test the assignment without the presence of the students. Assignments that do not contain this README with all the necessary instructions will not be evaluated. Assignments that do not execute correctly will also not be evaluated.

Students should upload the .zip file in the Inforestudante.

However, it is <u>mandatory</u> to deliver the report in the professor's locker up to 1 working day after the deadline stated above.

Good Work!