Projeto 1

Objetivo do programa

Nosso programa tem como objetivo ser um facilitador na hora dos professores orientadores escolherem os grupos e projetos de PFE, recebendo os alunos e suas preferencias de projetos, tendo em mente a maior quantidade de pessoas felizes nos grupos

Entradas

As entradas do programa tem um formato de competioções de programação para ser adequados a um programa em c, sendo a primeira linha de inputs o numero de alunos (n_alunos) projetos(n_projetos) e escolhas disponiveis(n_choices). E nas (n_alunos) linhas que seguem recebem os inputs (n_choices) separados por espaços para definir as escolhas dos alunos sobre a preferencia dos projetos

Saida

A saida final é um simples numero com a melhor metrica possivel de satisfacao juntamente com uma lista de como organizar os respectivos alunos nos respectivos projetos

Objetivo do projeto

Com este projetos comecamos com um programa python que resolvia nosso problema de uma forma ingenua sem se preocupar muito com eficiencia de condigo. Nosso objetivo é implementar uma maneira mais eficiente para executar o codigo alem de utilizarmos das tecnicas de parallelizacao ao nosso alcance, mas para isso precisamos primeiramente passar o nosso codigo para uma linguagem de mais baixo nivel e com uma preocupacao mais forte com eficiencia com o C++ que tem supporte ao OpemMP e trabalha com uma menor abstracao usando muitas vezes apis do kernel

Alem de nos proporcionar um caso real que pode acontecer para se utilizar paralelizacao e quais os seus modos corretos de uso em cada caso

Resultados

Em um primeiro momento decidimos fazer uma simples traducao de python para c++ para experenciar uma mudanca nos tempos de execucao e foi isso que fizemos como pode ser visto nos codigos backtrack.py backtrack.c bracktrack_arrat copy.cpp e backtrack.cpp alem de backtrack_local parallel.cpp e backtrack_no_omp

> podemos ver algumas semelhancas que se mantiveram no codigo

C++

```
if (i == n_alunos){
        if (melhor.satisfacao_atual==-1){
            //std::cout << aluno_projeto.data() << satisfacao_atual;
            melhor.satisfacao_atual = satisfacao_atual;
            melhor.melhor = aluno_projeto;

        }

        if (satisfacao_atual > melhor.satisfacao_atual){
            //std::cout << aluno_projeto;
            //std::cout << satisfacao_atual;
            melhor.satisfacao_atual = satisfacao_atual;
            melhor.melhor = aluno_projeto;
        }
        return melhor;


    }
```

```
std::vector<std::vector<int>> prefs(n_alunos);
    for ( int i = 0 ; i < n_alunos ; i++ )
        prefs[i].resize(n_projetos);



    for (int i = 0; i < n_alunos; i++)
    {
        int um, dois, tres;

        std::cin >> um;
        std::cin >> dois;
        std::cin >> tres;

        int prefs_choice[3] = {um, dois, tres} ;

        for (int j = 0; j < 3; j++)
        {
            prefs[i][prefs_choice[j]] = pow(n_choices - j, 2);

        }




    }


    std::vector<int> vagas (n_projetos,3); // start with value of 3

    std::vector<int> aluno_projeto (n_alunos,-1); // start with value of -1
```

Python

- Comparacao dos niveis de satisfacao

```
    if i == len(aluno_projeto): # todos alunos tem projeto
          if melhor is None:
              print('Melhor:', aluno_projeto, satisfacao_atual)
              melhor = aluno_projeto.copy(), satisfacao_atual
          if satisfacao_atual > melhor[1]:
              melhor = aluno_projeto.copy(), satisfacao_atual
              print('Melhor:', melhor)
          return melhor
```

- criacao de vetores

```
    for i in range(n_alunos):
          projs = [int(c) for c in input().split(' ')]
          for j, p in enumerate(projs):
              prefs[i, p] = pow(n_choices - j, 2)

      vagas = np.ones(n_projetos, np.uint) * 3 # 3 vagas por projeto
      aluno_projeto = np.ones(n_alunos, np.uint) * -1 # não escolheu pro
    jeto ainda
```

Tempos de execucao

Abaixo podemos ver os tempos de execucao usando a API de baixo nivel time do linux para cada um dos programas com diferentes inputs juntamente com graficos comparando-os

Entrada com 3 opcoes e 9 alunos (C++)

1 execucao

```
log
76
real    0m0,046s
user    0m0,042s
sys     0m0,004s
```

2 execucao

```
log
76
real    0m0,048s
user    0m0,044s
sys     0m0,004s
```

3 execucao

```
log
76
real    0m0,046s
user    0m0,045s
sys     0m0,000s
```

Entrada com 3 opcoes e 9 alunos (C++)

1 execucao

```
log
76
real    0m0,052s
user    0m0,041s
sys     0m0,003s
```

2 execucao

```
log
76
real    0m0,050s
user    0m0,043s
sys     0m0,002s
```

3 execucao

```
log
76
real    0m0,051s
user    0m0,040s
sys     0m0,000s
```

Entrada com 3 opcoes e 9 alunos (C++)

1 execucao

```
log
76
real    0m3,100s
user    0m0,041s
sys     0m0,013s
```

2 execucao

```
log
76
real    0m2,720s
user    0m0,043s
sys     0m0,002s
```

3 execucao

```
log
76
real    0m2,850s
user    0m0,030s
sys     0m0,008s
```

Entrada com 3 opcoes e 9 alunos (Python)

1 execucao

```
log
76
real    0m0,208s
user    0m0,608s
sys     0m0,497s
```

2 execucao

```
log
76
real    0m0,207s
user    0m0,642s
sys     0m0,459s
```

3 execucao

```
log
76
real    0m0,213s
user    0m0,567s
sys     0m0,539s
```

Entrada com 2 opcoes e 6 alunos (Python)

1 execucao

```
log
21
real    0m0,182s
user    0m0,509s
sys     0m0,411s
```

2 execucao

```
log
21
real    0m0,176s
user    0m0,525s
sys     0m0,458s
```

3 execucao

```
log
21
real    0m0,178s
user    0m0,524s
sys     0m0,426s
```

Entrada com 2 opcoes e 6 alunos (Python)

1 execucao

```
log
103
real    0m11,880s
user    0m12,317s
sys     0m0,481s
```

2 execucao

```
log
103
real    0m10,447s
user    0m10,889s
sys     0m0,481s
```

3 execucao

```
log
103
real    0m10,684s
user    0m11,123s
sys     0m0,488s
```

Menor entrada (C++ / Local)

1 Execucao

```
log
66
real    0m0,005s
user    0m0,004s
sys     0m0,000s
```

2 Execucao

```
log
73
real    0m0,008s
user    0m0,001s
sys     0m0,007s
```

3 Execucao

```
68
real    0m0,008s
user    0m0,005s
sys     0m0,004s
```

entrada Media (C++ / Local)

1 Execucao

```
log
58
real    0m0,007s
user    0m0,007s
sys     0m0,000s
```

2 Execucao

```
log
58
real    0m0,008s
user    0m0,001s
sys     0m0,007s
```

3 Execucao

```
61
real    0m0,008s
user    0m0,005s
sys     0m0,004s
```

Entrada Grande (C++/Local)

1 Execucao

```
log
80
real    0m0,008s
user    0m0,008s
sys     0m0,000s
```

2 Execucao

```
log
80
real    0m0,009s
user    0m0,001s
sys     0m0,009s
```

3 Execucao

```
log
77
real    0m0,011s
user    0m0,011s
sys     0m0,000s
```

Menor entrada (C++ / Extensiva / Paralella)

1 Execucao

```
log
76
real    0m0,024s
user    0m0,121s
sys     0m0,004s
```

2 Execucao

```
log
76
real    0m0,024s
user    0m0,122s
sys     0m0,007s
```

3 Execucao

```
76
real    0m0,041s
user    0m0,230s
sys     0m0,000s
```

entrada Media (C++ / Extensiva / Paralella)

1 Execucao

```
log
42
real    0m0,041s
user    0m0,206s
sys     0m0,019s
```

2 Execucao

```
log
42
real    0m0,043s
user    0m0,248s
sys     0m0,004s
```

3 Execucao

```
42
real    0m0,042s
user    0m0,218s
sys     0m0,016s
```

Entrada Grande (C++ / Extensiva / Paralella)

1 Execucao

```
log
103
real    0m1,013s
user    0m6,414s
sys     0m0,037s
```

2 Execucao

```
log
103
real    0m0,998s
user    0m6,617s
sys     0m0,036s
```

3 Execucao

```
log
103
real    0m1,011s
user    0m6,686s
sys     0m0,068s
```

Menor entrada (C++ / local / Paralella)

1 Execucao

```
log
76
real    0m0,018s
user    0m0,083s
sys     0m0,005s
```

2 Execucao

```
log
76
real    0m0,032s
user    0m0,168s
sys     0m0,001s
```

3 Execucao

```
76
real    0m0,040s
user    0m0,227s
sys     0m0,004s
```

entrada Media (C++ / Extensiva / Paralella)

1 Execucao

```
log
66
real    0m0,036s
user    0m0,169s
sys     0m0,009s
```

2 Execucao

```
log
66
real    0m0,036s
user    0m0,169s
sys     0m0,009s
```

3 Execucao

```
66
real    0m0,036s
user    0m0,169s
sys     0m0,009s
```

Entrada Grande (C++ / Extensiva / Paralella)

1 Execucao

```
log
9
real    0m0,032s
user    0m0,172s
sys     0m0,008s
```

2 Execucao

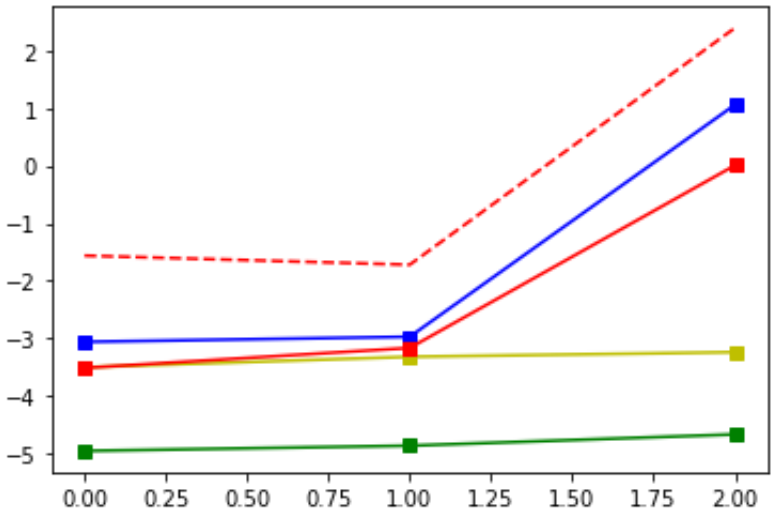
```
log
9
real    0m0,042s
user    0m0,226s
sys     0m0,004s
```

3 Execucao

```
log
9
real    0m0,043s
user    0m0,255s
sys     0m0,001s
```

In [3]:

```
import matplotlib.pyplot as aaa
import numpy as np

##Entrada peqiena
python1 = [.208, .207, .213]
cpp1 = [.046, .048, .046]
cpp1_local= [.005, .008, .008]
cpp1_para = [.024, .024, .041]
cpp1_local_para= [.018, .032, .040]

##entrada media
python2 = [.182, .176, .178]
cpp2 = [.052, .050, .051]
cpp2_local= [.007, .008, .008]
cpp2_para= [.041, .043, .042]
cpp2_local_para= [.036, .036, .036]

#Entrada Grande
python3 = [11.880, 10.447, 10.684]
cpp3 = [3.10, 2.72, 2.85]
cpp3_local= [.008, .009, .011]
cpp3_para= [1.013, .998, 1.011]
cpp3_local_para= [.032, .042, .043]


pythonn = [np.mean(python1), np.mean(python2), np.mean(python3)]
cpp = [np.mean(cpp1), np.mean(cpp2), np.mean(cpp3)]
cpp_local = [np.mean(cpp1_local), np.mean(cpp2_local), np.mean(cpp3_l
ocal)]
cpp_local_para = [np.mean(cpp1_local_para), np.mean(cpp2_local_para),
np.mean(cpp3_local_para)]
cpp_para =[np.mean(cpp1_para), np.mean(cpp2_para), np.mean(cpp3_para)
]

ii = range(3)

aaa.plot(ii, np.log(pythonn), 'r--', ii, np.log(cpp), 'bs-', ii, np.l
og(cpp_local), 'gs-', np.log(cpp_local_para), 'ys-', np.log(cpp_para)
, 'rs-')
aaa.show()
#-- vermelha -> Python
# azul -> C++
# verde -> cpp local
# amarelo -> cpp local para
# vermelho cpp_paralelo
```



Conclusao

Como podemos ver, sem o uso das tecnicas de paralelizacao tivemos um aumento na eficiencia de 70 % em todos os testes em media as tecnicas de paralelizacao funcionam muito bem para as tecnicas que utilizam da busca local pelo fato de utilizar um loop que pode muito bem ser paralelizado mas ter permanecido mais ou menos na mesma para a paralelizacao de recursividade mesmo mostrando um desempenho pouco superiror