Um *Framework* de Otimização para Aplicações de Alto Desempenho em *Cloud*

Danilo Oliveira

Orientador: Paulo Maciel
Co-orientador: Nelson Rosa

Introdução

O que é a High Performance Computing

e qual a relação dela com a *cloud computing*?

Introdução

Qual a importância da tolerância a falhas em aplicações de HPC?

Introdução

- Considere o seguinte cenário:
  - 20,000 processadores
  - Cada um com MTTR de 5 anos
  - Porém, todos são essenciais para a aplicação

Introdução

- Considere o seguinte cenário:
  - 20,000 processadores
  - Cada um com MTTR de 5 anos
  - Porém, todos são essenciais para a aplicação
- Resultado: O MTTF do sistema é de menos de 1 hora!!!

Técnicas de tolerância a falhas em HPC

- Checkpointing e Restart
- Processadores redundantes

Proposta

O objetivo deste trabalho é criar um Framework para suportar o planejamento e otimização do desempenho de aplicações em HPC em nuvem, considerando falhas nos nós virtualizados e estratégias de redundância temporal e espacial. Este Framework deverá permitir que um usuário de uma nuvem computacional que deseje submeter uma aplicação de alto desempenho saiba estimar o tempo e o valor a ser pago ao provedor, sendo que a alocação dos recursos em nuvem e configuração da aplicação será otimizada de forma a reduzir estas duas métricas

Proposta

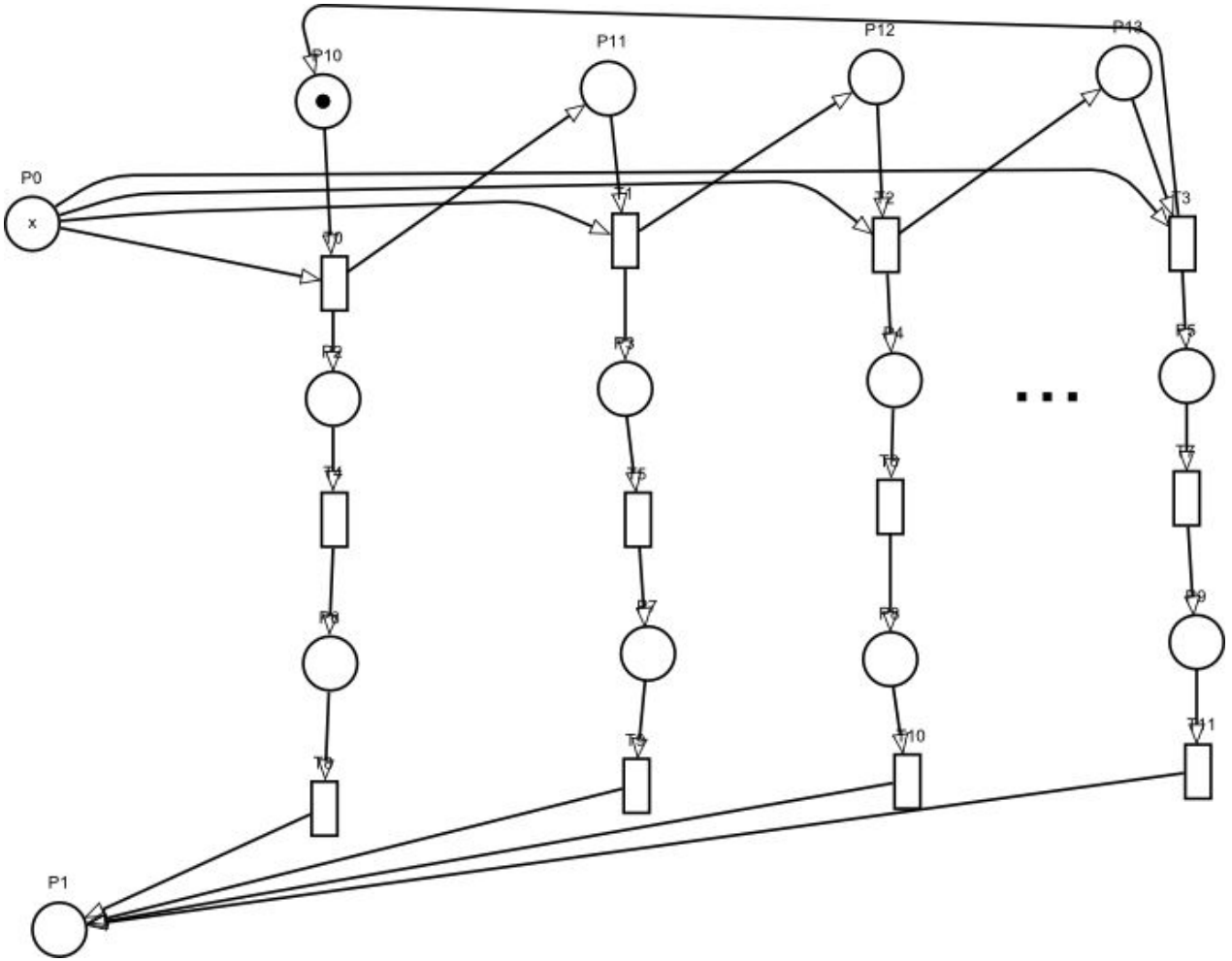
- Extensões na linguagem MSL para modelar uma aplicação de HPC com arquitetura mestre escravo:
  - Comandos de repetição e condicional na declaração de modelos

Exemplo (Pseudo código MPI):

```
if( rank = 0 ) { // master
    int r = 1;
    for( int i = 0; i < numTasks; i++ ) {
        send task i to processor rank r;

        r = if (r == NUM_PROCESSORS) then 1 else r+1;
    }

}else{
    receive task;
    process task;
    send result back to master;
}
```


P10
P11
P12
P13
P0
x
T0
T1
T2
T3
P2
P3
P4
P5
T4
T5
T6
T7
P6
P7
P8
P9
T8
T9
T10
T11
P1
...

```
7  SPN Model{
8
9       place tasks( tokens = ntasks );
10      place completedtasks;
11
12
13
14      for i in range(1, nprocs){
15
16          if( $i == 1 ){
17              place p_#($i)(tokens = 1);
18          }else{
19              place p_#($i);
20          }
21
22          place inbuffer_#($i);
23          place outbuffer_#($i);
24      }
```

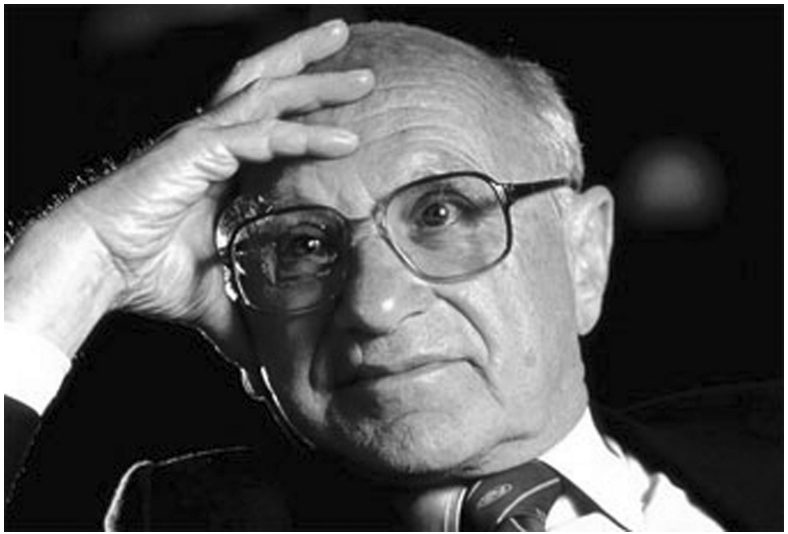
```
for i in range(1, nprocs){
     timedTransition process_#($i)(
          inputs = [inbuffer_#($i)],
          outputs = [outbuffer_#($i)],
          delay = proc
     );

     if($i != nprocs){
          timedTransition recv_#($i)(
               inputs = [tasks, p_#($i)],
               outputs = [inbuffer_#($i), p_#(i+1)],
               delay = recv
          );
     }else{
          timedTransition recv_#($i)(
               inputs = [tasks, p_#($i)],
               outputs = [inbuffer_#($i), p_#(1)],
               delay = recv
          );
     }

     timedTransition send_#($i)(
          inputs = [outbuffer_#($i)],
          outputs = [completedtasks],
          delay = send
     );
}
```

Problemas com essa abordagem

“Não existe almoço grátis em modelos baseados em espaço de estados”



Conclusões

- O presente trabalho irá contribuir para a comunidade da computação de alto desempenho provendo modelos que se situam em um meio termo entre modelos analíticos e modelos de simulação tradicionais
- Os próximos passos serão a modelagem de problemas mais complexos, modelagem de mecanismos de redundância, experimentos de validação, e aplicação de técnicas de otimização e análise de sensibilidade com o objetivo de encontrar a configuração ótima de parâmetros para cada problema de forma customizada