

# Modelo de Gerenciamento de Nível de Serviço baseado na Qualidade de Serviço Esperada pelo Usuário e aplicado em Aplicações Distribuídas suportadas por uma plataforma *Grid*.

Solange Sari

[solange, elvis}@csd.uwo.ca](mailto:{solange, elvis}@csd.uwo.ca)

Elvis Vieira\*

Paulo Freitas Filho

Universidade Federal de Santa Catarina

Departamentos INE/EPS

Florianópolis – SC - Brasil

Michael Bauer, Hanan Lutffiyya

The University of Western Ontario

Department of Computer Science

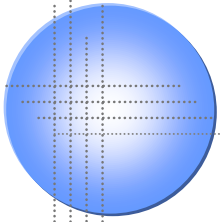
London – ON - Canada

\* CNPq 200323/01-6



# Agenda

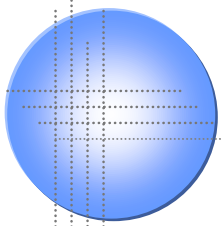
- Objetivos;
- Infraestrutura Grid;
- Gerenciamento de Serviço Grid;
- Solução Proposta,
  - Algoritmo de escalonamento;
- Implementação,
  - Escalonador,
  - Simulador,
  - Gerente;
- Exemplo passo a passo.



# Objetivo Geral

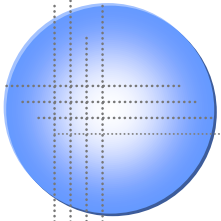
- Introduzir a qualidade de serviço esperada pelo usuário em aplicações distribuídas numa plataforma Grid.





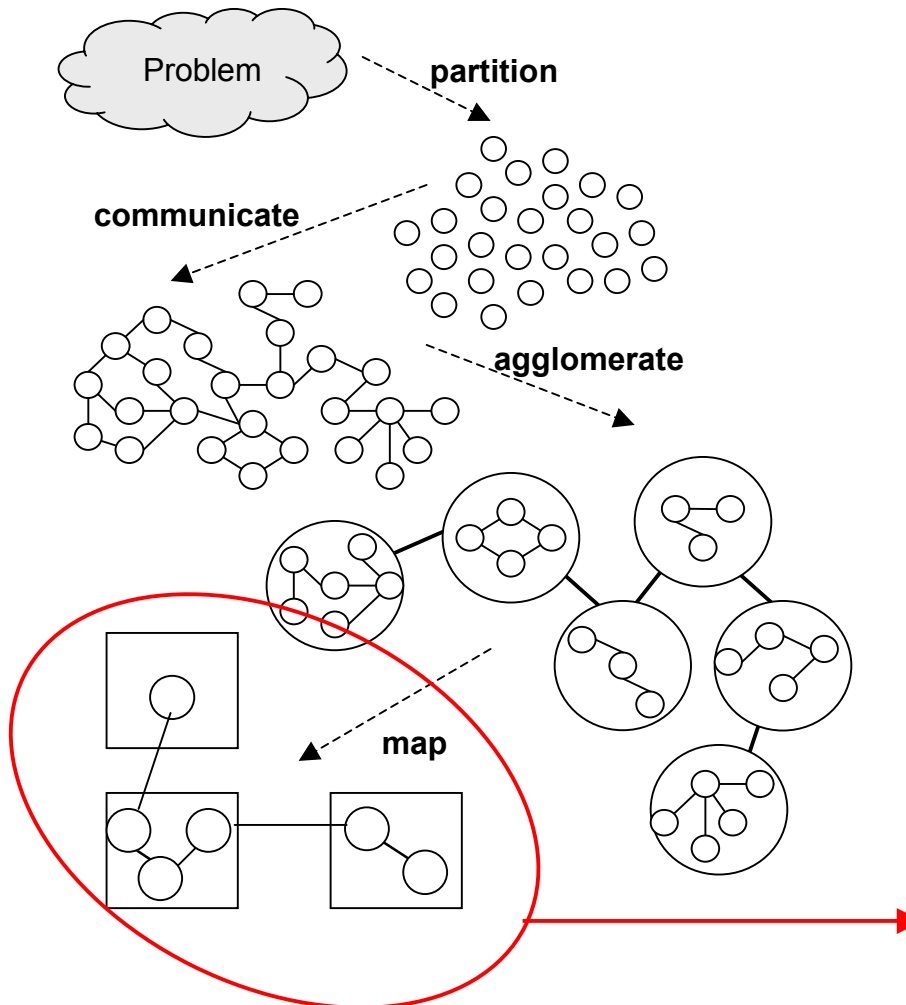
# Aplicações Grid

- Centradas na Comunidade,
  - Access Grid;
- Centradas em Dados,
  - Data Grid;
- Centradas em Computação,
  - Grid Computing;
- Centradas em Interação,
  - (real time interaction).



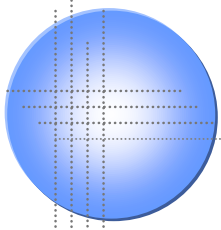
# Problemas de Metacomputação

## Método PCAM



**Escalonamento**

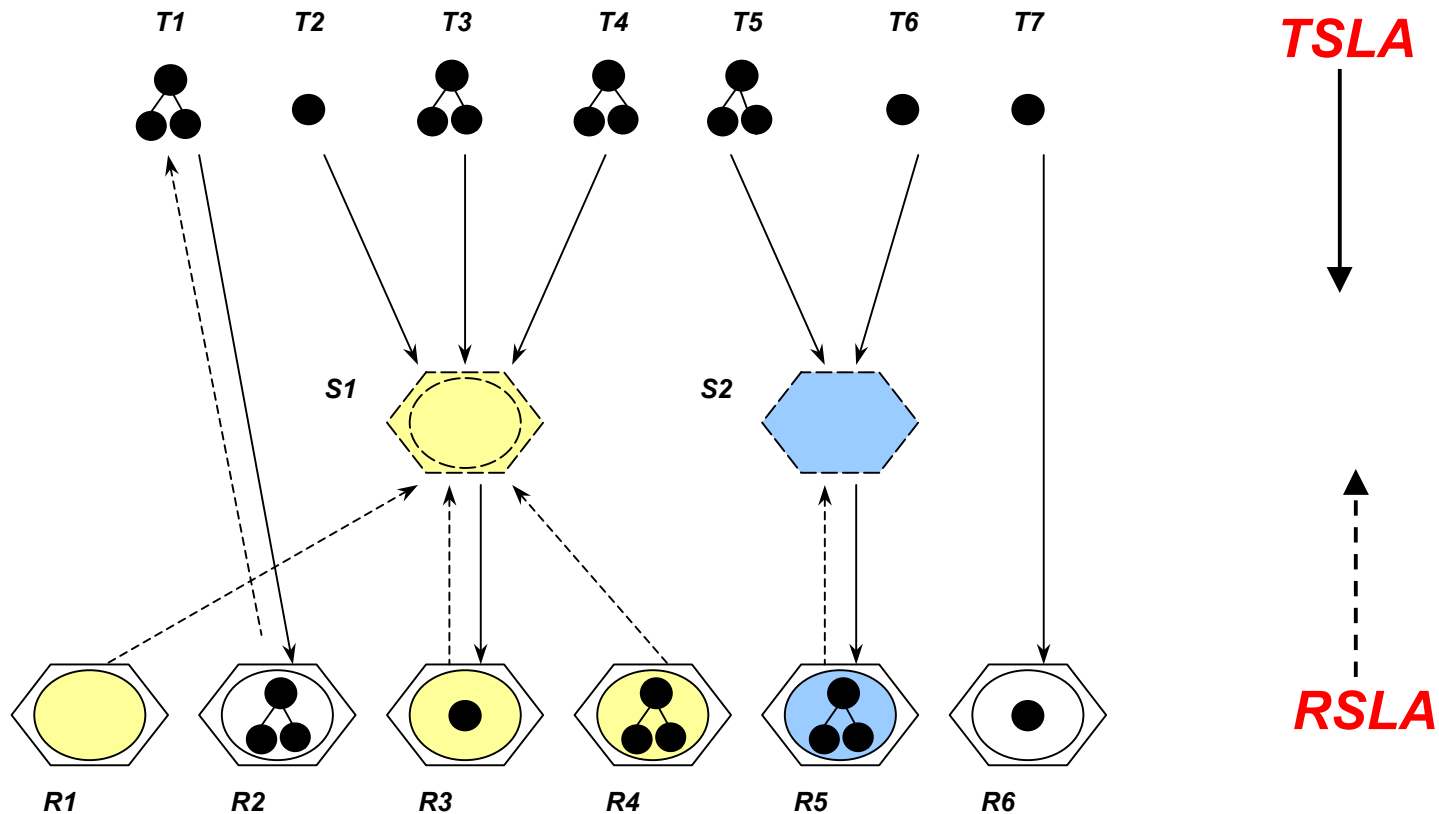




# Gerenciamento de Nível de Serviço

## ■ Cenário de Escalonadores Comunitários.

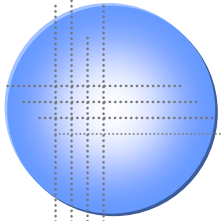
- SNAP (*Service Negotiation and Acquisition Protocol*)
  - \* GRAAP (*Grid Resource Allocation Agreement Protocol*)



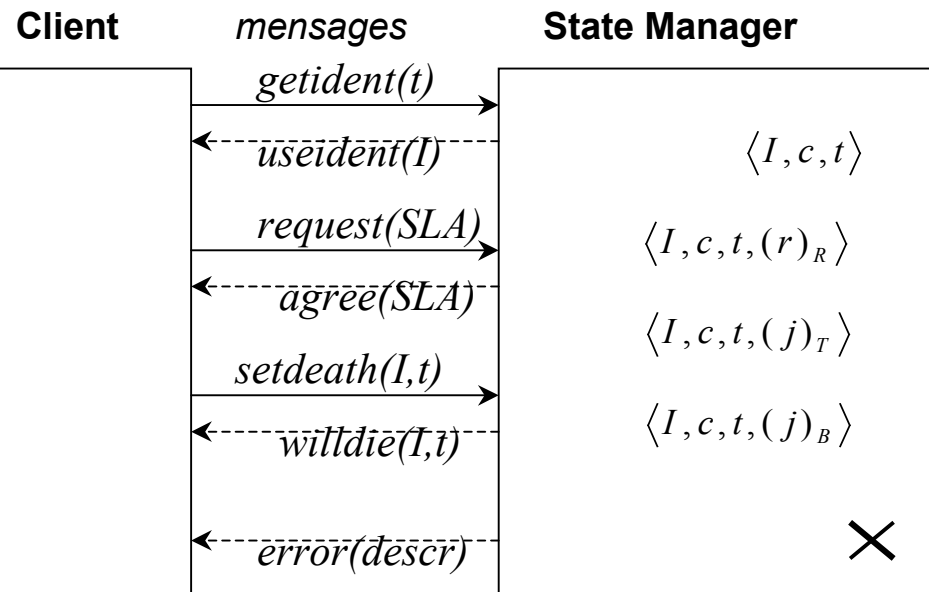
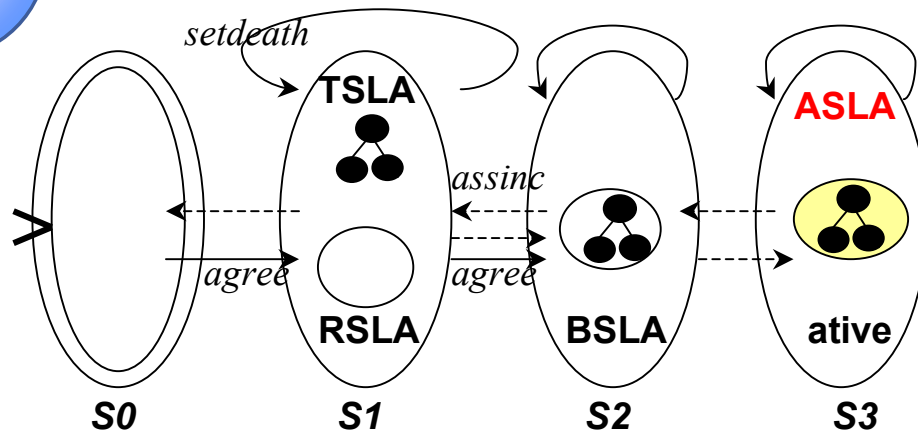
# Acordos de Níveis de Serviços

- TSLA - *Task Service Level Agreements*;
  - $\langle I, c, t_{\text{dead}}, (j)_T \rangle$ ;
- RSLA - *Resource Service Level Agreement*
  - $\langle I, c, t_{\text{dead}}, (r)_R \rangle$ ;
- BSLA - *Binding Service Level Agreement*
  - $\langle I, c, t_{\text{dead}}, (j)_B \rangle$ ; and
- ASLA - *Adapted Service Level Agreement*,
  - $\langle I, c, t_{\text{dead}}, (j)_A \rangle$ .

SLA's Descriptions?



# Controle de SLAs

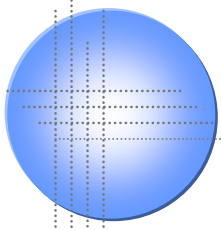




# Solução Proposta

---

- Aplicar a abordagem de escalonamento a nível de aplicação no cenário de escalonadores comunitários.
  - Usando na heurística o modelo de qualidade de serviço.
  
- Baseado no agente AppLeS .
  - <http://grail.sdsc.edu>



# Algoritmo – Scheduler( )

Scheduler( $\Gamma$ : Task List,  $\Pi$  : Resource List )

{

*Process information in*  $\Gamma$  and  $\Pi$ ;

while

foreach *task*  $i$

$Q_t = [t_i, t_s]$

*getident*( $t_{dead}$ ) *useident*( $I, t_{dead}$ )

*Create* TSLA[(processor,  $t$ ,  $Q_p[a^+, r_i, r_s]$ ),  
(memory,  $q$ ,  $Q_m[a^-, r_i, r_s]$ ),  
(link,  $t$ ,  $Q_l[a^+, r_i, r_s]$ ),  
(disk,  $q$ ,  $Q_d[a^-, r_i, r_s]$ )

*request*( $I, i, t_{dead}, TSLA$ ) *agree*( $I, i, t_{dead}, TSLA'$ )

*Apply the function* Heuristic( )

endforeach

*Apply the function* Simulator( )

Update  $\Gamma$  and  $\Pi$ ;

*Apply the function* Manager( )

endwhile

}



# Algoritmo– Heuristic( ) - 1

```
Heuristic()  
{  
  foreach domain j  
    foreach nodo n  
      foreach processor k  
        getident(tdead) useident(I, tdead)  
         $P_{i,j,n,k} = Q(T_i, P_{j,n,k})$   
        if  
           $P_{i,j,n,k} > t_s$   
          Available( )  
        endif  
        Gera RSLA[processor, pi,j,k]  
        request(I, i, tdead, RSLA) agree(I, i, tdead, RSLA')  
      endfor  
       $P_{i,j,n} = \max_k \{P_{i,j,n,k}\}$ 
```

# Algoritmo– Heuristic( ) - 2

$$c_{i,j,n} = \frac{\partial_1 * p_{i,j,n} + \partial_2 * m_{i,j,n} + \partial_3 * l_{i,j,n} + \partial_4 * d_{i,j,n}}{\partial_1 + \partial_2 + \partial_3 + \partial_4}$$

```
endeach  
while  
     $c_{i,j} \notin Q_t$   
     $c_{i,j} = \max_n (c_{i,j,n})$   
     $n + 1$   
endwhile  
 $c_i = \max_j (c_{i,j})$   
 $task\ i \rightarrow c_i$   
 $getident(t_{dead})\ useident(I, t_{dead})$   
Gera BSLA[(processor,  $p_i$ ),  
          (memory,  $m_i$ ),  
          (link,  $l_i$ ),  
          (disk,  $d_i$ )]  
 $request(I, i, t_{dead}, BSLA)\ agree(I, i, t_{dead}, BSLA')$   
endeach  
}
```

# Algoritmo – Manager( )

**Manager( )**

*getident*( $t_{\text{dead}}$ ) *useident*( $I, t_{\text{dead}}$ )

→ Gera ASLA[(processor,  $\rho_i$ ),  
(memory,  $\mu_i$ ),  
(link,  $\lambda_i$ ),  
(disk,  $\delta_i$ )]

*request*( $I, i, t_{\text{dead}}, \text{ASLA}$ ) *agree*( $I, i, t_{\text{dead}}, \text{ASLA}'$ )

$$b = \frac{\partial_1 p + \partial_2 m + \partial_3 l + \partial_4 d}{\partial_1 + \partial_2 + \partial_3 + \partial_4} \quad a = \frac{(\partial_1 \rho + \partial_2 \mu + \partial_3 \lambda + \partial_4 \delta)}{\partial_1 + \partial_2 + \partial_3 + \partial_4}$$

**if**

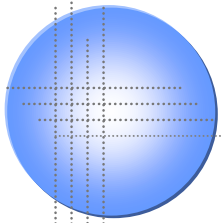
→  $|a - b| > 0.1 \wedge a \notin Q_t$

**then**

$F = F + 1$

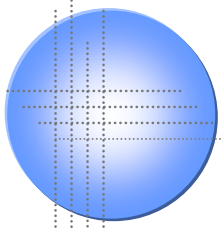
**endif**

}

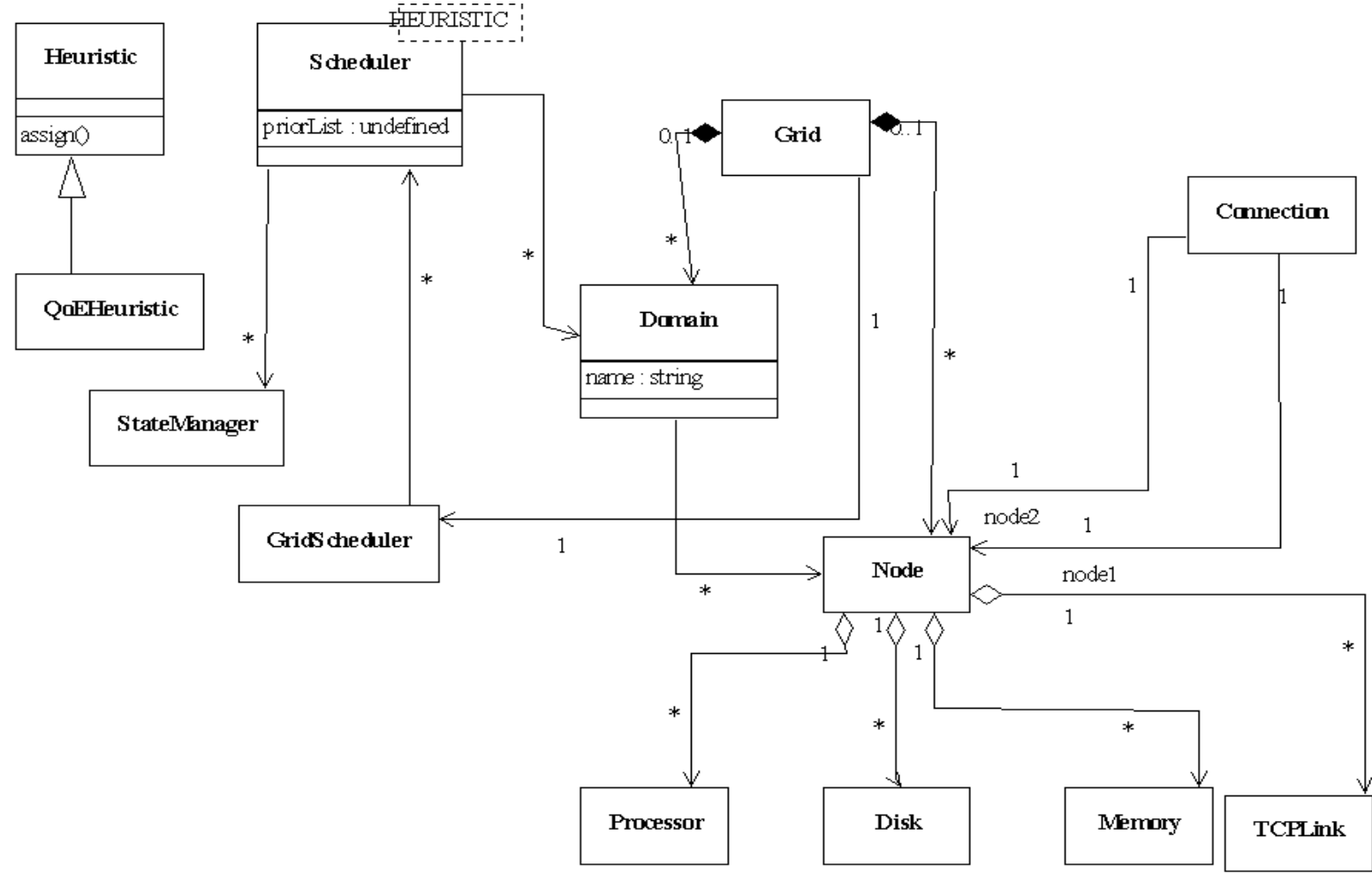


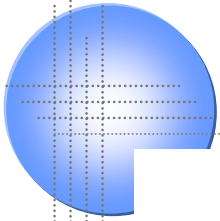
# Implementação

- Linguagem C++;
- Biblioteca XERCESC++ (leitura XML);
- Biblioteca SimGrid – C (simulação);
- Bibliotecas Auxiliares :
  - Standard Library, BOOST Library;
- Compilador da GNU para Linux.

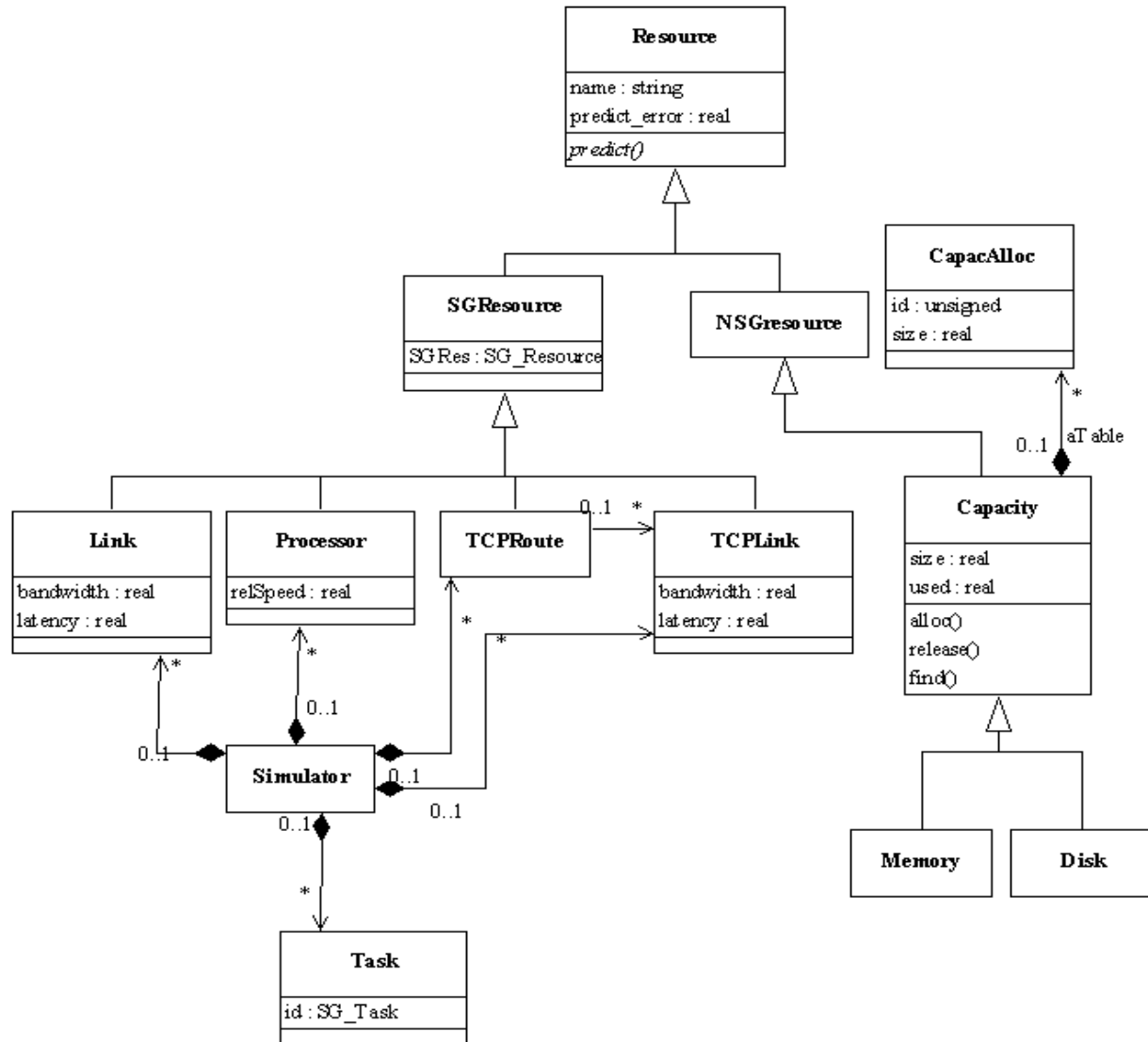


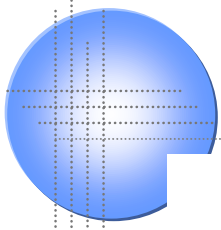
# Escalonador Grid



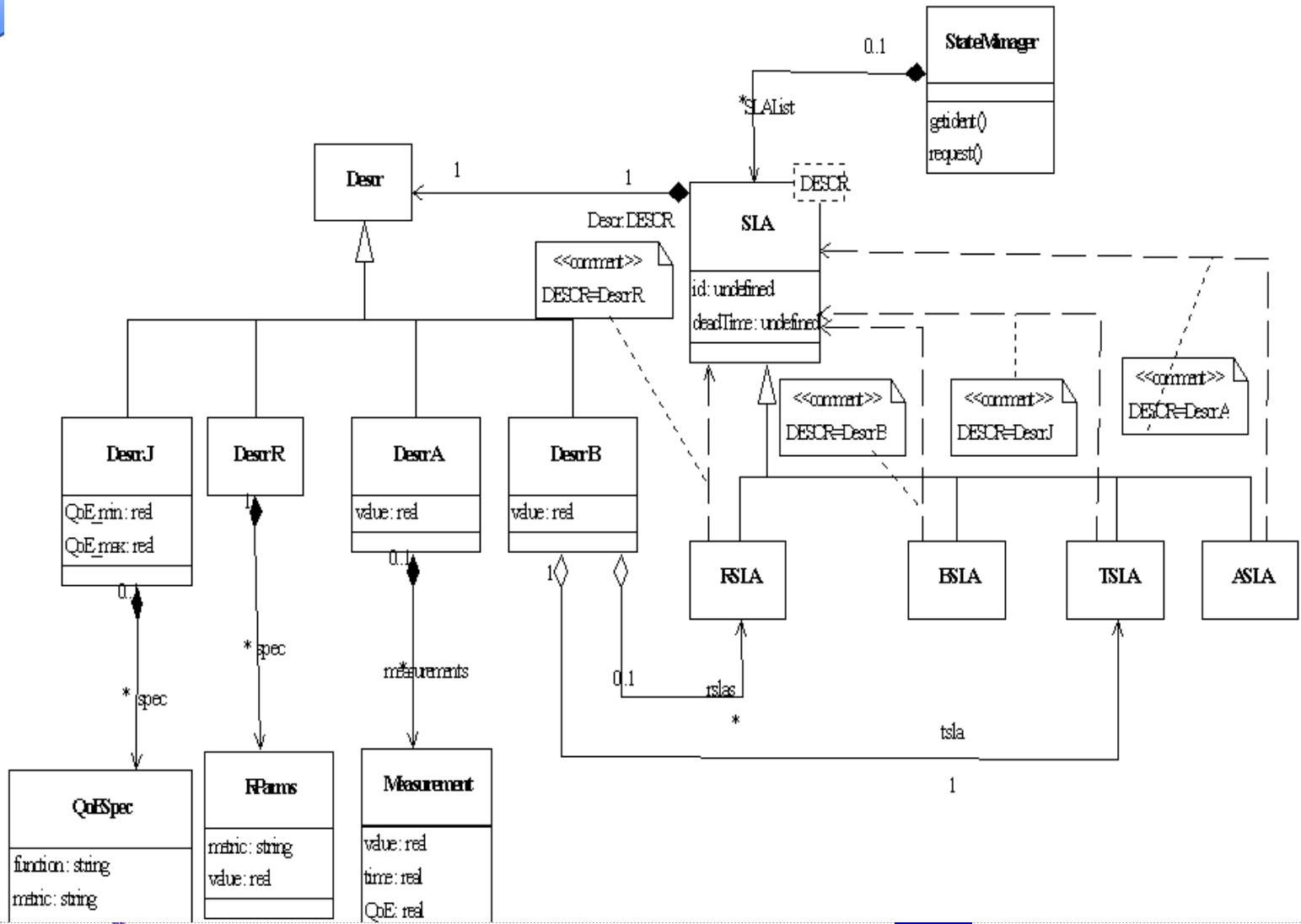


# Simulador



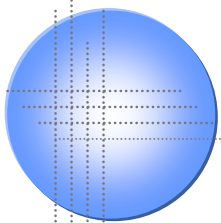


# Gerente



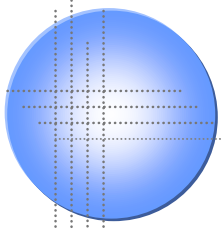
# Descrição dos TSLAs

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Master/Slave Application -->
<SLA xmlns=www.csd.uwo.ca/gridsla C:\XML\sla.xsd" title="String"
description="String" model="MASTER_WORKERS">
<TASK name="A" id="000001">
  <DEAD_TIME duration="600" unit="s"/>
  <QOE_TASK min_level="0.0" max_level="0.8"/>
    <REQUEST name="CPU_TIME">
      <QOE_RESOURCE function="Qp">
        <LEVEL name="rs" value="35.00" unit="s"/>
        <LEVEL name="ri" value="25.00" unit="s"/>
        <LEVEL name="t" value="50" unit="integer"/>
        <LEVEL name="a" value="-0.275" unit="integer"/>
        <LEVEL name="wp" value="1.0" unit="integer"/>
      </QOE_RESOURCE>
    </REQUEST>
    <REQUEST name="MEMORY">
      <QOE_RESOURCE function="Qm">
        .....
      </QOE_RESOURCE>
    </REQUEST>
    <REQUEST name="TRANSFER_SIZE">
      <QOE_RESOURCE function="Ql">
        .....
      </QOE_RESOURCE>
    </REQUEST>
    <REQUEST name="DISK_SPACE">
      <QOE_RESOURCE function="Qd">
        .....
      </QOE_RESOURCE>
    </REQUEST>
  <PRECEDENCE cost="0" parent="000001"/>
</TASK>
```



# Descrição dos RSLAs

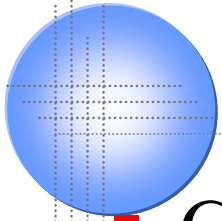
```
<?xml version="1.0" encoding="iso-8859-1"?>
<GRID>
  <LABEL name="GridUWO" />
    <SITE domain="spinner.sharcnet.ca">
      <LABEL name="spinner" />
        <MACHINE>
          <LABEL ip="192.168.7.1" name="sp1.spinner.cluster">
            <ALIAS name="sp1" /></LABEL>
            <PROPERTY name="CPU_RSPEED" value="6" />
            <PROPERTY name="MEMORY" value="1500.0" units="MB"/>
            <PROPERTY name="DISK_SPACE" value="3000.0" units="MB" />
          </MACHINE>
          <MACHINE>
            .....
            </MACHINE>
            <LINK node1="sp6.spinner.cluster" node2="default">
              <LABEL ip="192.168.7.6" name="sp1-6">
                <ALIAS name="sp1-sp6" /></LABEL>
                <PROPERTY name="BANDWIDTH" value="300.00" units="Mbs" />
                <PROPERTY name="LATENCY" value="0.0008" units="s" />
              </LINK>
              <LINK node1="spx.spinner.cluster" node2="default">
                .....
                </LINK>
            </SITE>
            <SITE domain="syslab.csd.uwo.ca">
              .....
            </SITE>
          </GRID>
```



# Exemplo – Passo 1

- 1. Especificar os recursos requeridos:

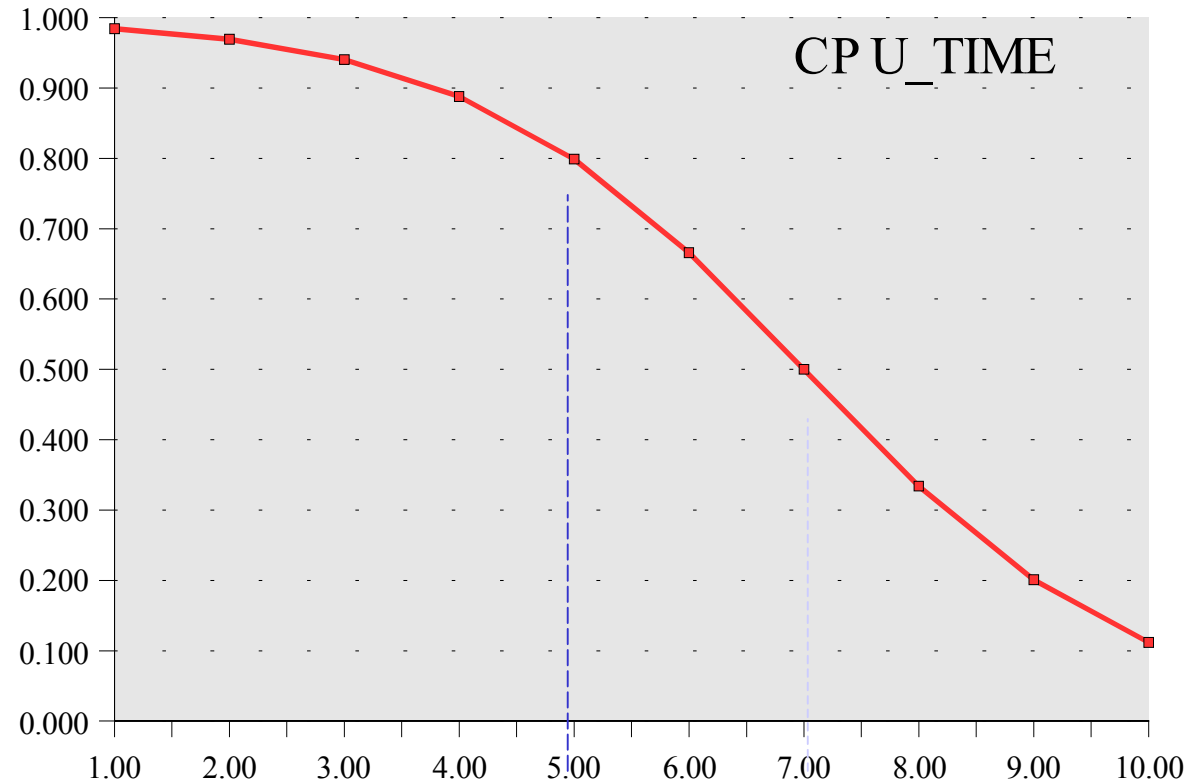
	W(u)	Q(Mb)	Memory (MB)	Disk (MB)	Ti	Ts
A	50	1500.00	1400	1500	0.50	0.80
B	10	300.00	400	500	0.50	0.80
C	10	300.00	400	500	0.50	0.80
D	10	300.00	400	500	0.50	0.80
E	10	300.00	400	500	0.50	0.80
F	10	300.00	400	500	0.50	0.80
	Wp	Wl	Wm	Wd		
	1.0	1.0	1.0	1.0		



## Exemplo – Passo 2

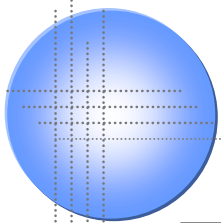
■ Configurar a Função de QoE:  $Q_c^a(x) = \frac{1}{1 + e^{-a(x-c)}}$

CPU_TIME	
a	0.690
C	7.000
1.00	0.984
2.00	0.969
3.00	0.940
4.00	0.888
5.00	0.799
6.00	0.666
7.00	0.500
8.00	0.334
9.00	0.201
10.00	0.112



CPU\_TIME = W(u) / CPU\_RSPEED

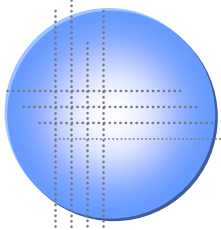




## Exemplo – Passo 3

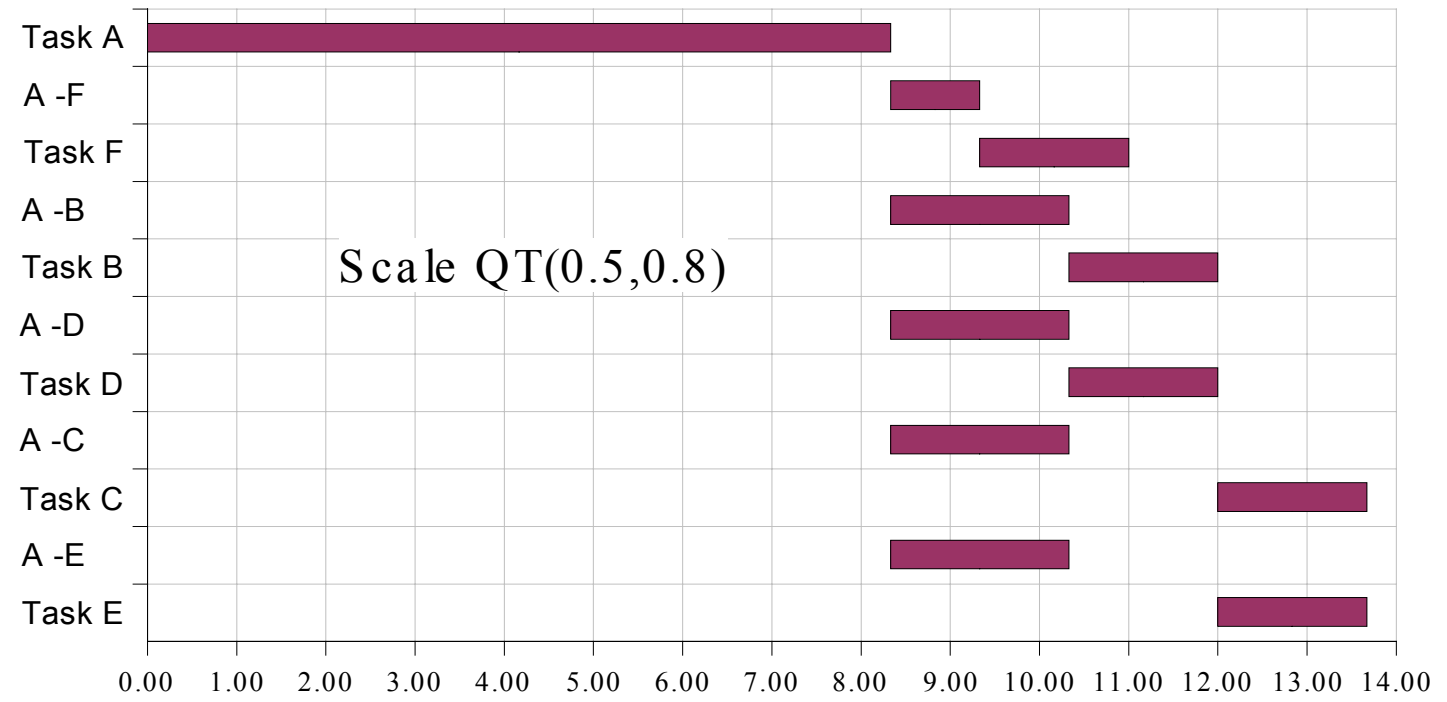
- Especificar a disponibilidade dos recursos:

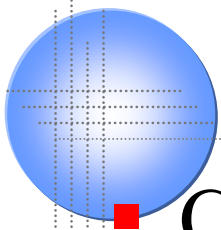
domain	node	name	cpu_rspeed (u)	Bandwith (Mbps)	Latency (s)	Memory (MB)	Disk (MB)
0	0	sp1	6	300.00	0.0008	1500.00	3000.00
0	1	sp2	6	300.00	0.0008	1500.00	3000.00
0	2	sp3	6	300.00	0.0008	1500.00	3000.00
0	3	sp4	6	300.00	0.0008	1500.00	3000.00
0	4	sp5	6	300.00	0.0008	1500.00	3000.00
0	5	sp6	6	300.00	0.0008	1500.00	3000.00
1	0	durendal	5	200.00	0.0010	1000.00	2000.00
1	1	montpurse	5	200.00	0.0010	1000.00	2000.00
1	2	doublefudge	6	300.00	0.0008	1500.00	3000.00
1	3	vanilla	6	300.00	0.0008	1500.00	3000.00
1	4	peppermint	7	400.00	0.0006	2000.00	4000.00
1	5	scoop	7	400.00	0.0006	2000.00	4000.00



# Exemplo – Passo 4

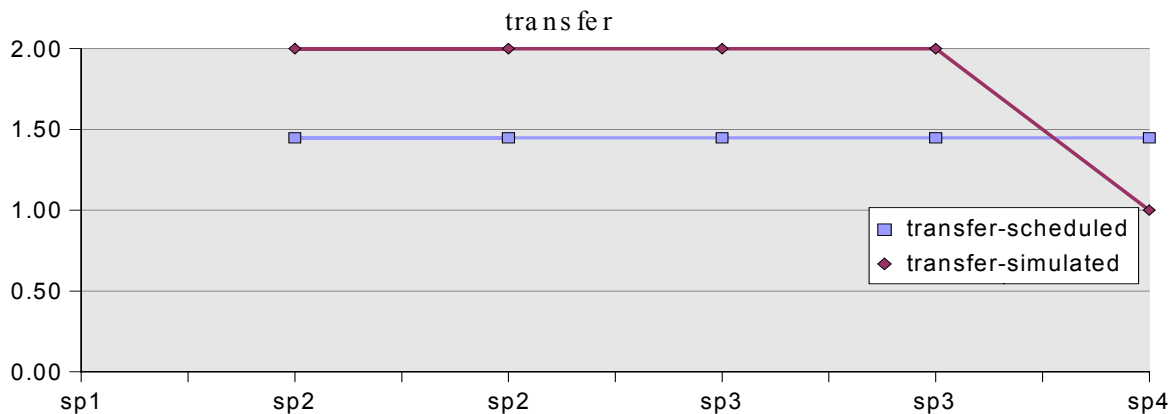
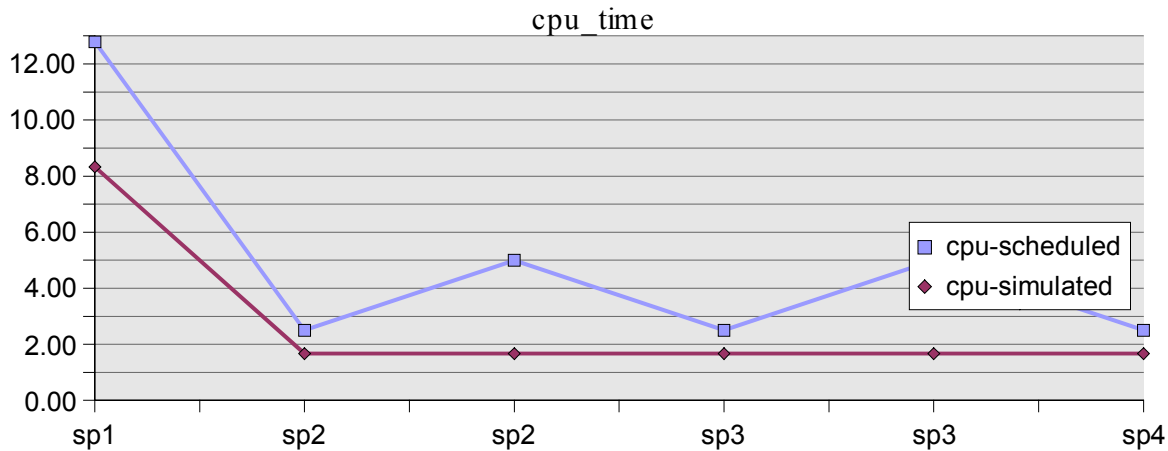
## ■ Simular tarefas e recursos:





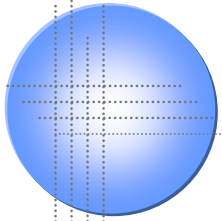
# Exemplo – Passo 5

■ Comparar tempo escalonado e simulado.



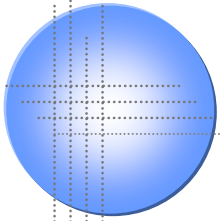
# Conclusões

- O desempenho da aplicação depende do **modelo da aplicação**;
  - O modelo de gerenciamento proposto aumenta a responsabilidade do projetista da aplicação ao estabelecer níveis de serviço.
- O desempenho da aplicação depende da **heurística de escalonamento**;
  - A heurística proposta utiliza a qualidade de serviço de vários recursos para a tomada de decisão;
  - A heurística proposta garante satisfazer as necessidades do usuário, mas não garante o melhor desempenho em termos de tempo de execução.

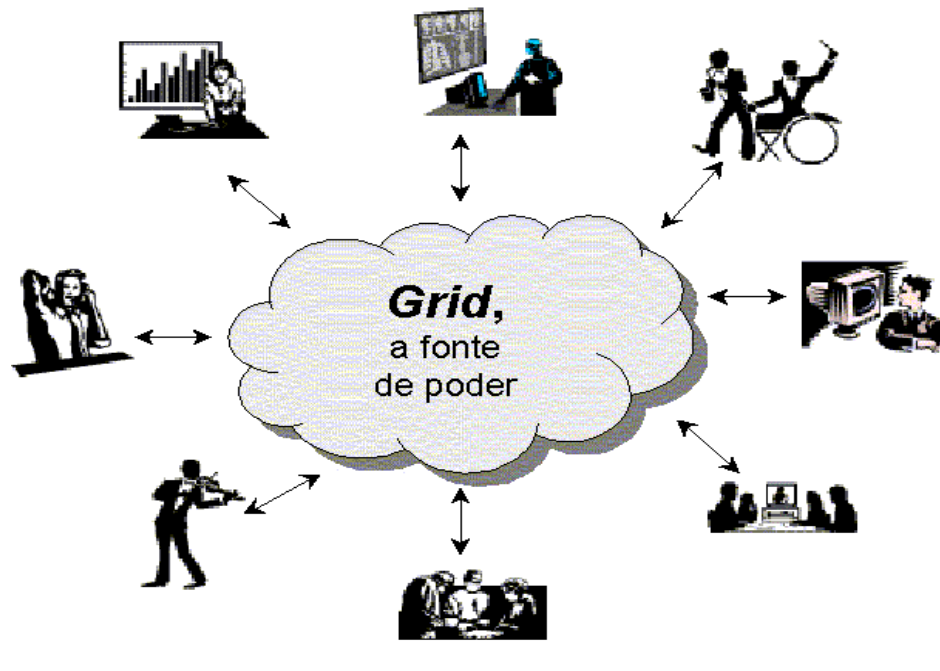


# Trabalhos Futuros

- A solução de implementação (escalonador + simulador + gerente) deve ser aplicado a problemas reais;
  - Descrição de tarefas de aplicações padrões;
  - Descrição de recursos dinâmica;
- O gerente de níveis de serviço pode interferir na negociação;
- Aplicar modelo no escalonamento feito pelo sistema de gerenciamento de recursos.



# Questões ?



Modelo de Gerenciamento de Nível de Serviço baseado na Qualidade de Serviço Esperada pelo Usuário e aplicado em Aplicações Distribuídas suportadas por uma plataforma *Grid*.

Solange Sari

Elvis Vieira

{solange, elvis}@csd.uwo.ca

