Um Estudo sobre as Propriedade de Vivacidade no MobiGrid

Aluno: Rodrigo Moreira Barbosa
Orientador: Prof. Dr. Alfredo Goldman vel Lejbman
Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

Roteiro da Apresentação

1. Motivação
2. Visão geral do MobiGrid
3. Experimentos com clonagem e migração
4. Modelo matemático da simulação
5. Experimentos com a simulação
6. Referências

1. Motivação - Projetos

- O InteGrade está criando uma infra-estrutura de middleware que permite a utilização de recursos ociosos.
- O MobiGrid é um projeto para a criação de uma infra-estrutura de agentes móveis para um ambiente de grade.
- MobiGrid provê um ambiente de programação para aplicações com longo tempo de processamento encapsuladas em um agente móvel (*tarefas*).

1. Motivação
1.1 Agentes Móveis

- Agentes móveis permitem:
  - transparência em termos de performance para o usuário local;
  - os recursos ociosos podem ser usados da melhor maneira possível.
- Opções para um agente móvel quando seu recurso não está mais disponível:
  - migração;
  - terminação;
  - suspensão.

1. Motivação
1.2 Estratégia do MobiGrid

- Múltiplas cópias de uma *tarefa*:
  - solução simples e leve;
  - também uma estratégia de recuperação de erro.
- *Vivacidade*:
  - propriedade de uma *tarefa* que tem mais de uma cópia sendo executada;
  - ambiente de controle da *vivacidade* com clonagem e migração.

1. Motivação
1.3 Simulação

- Comparação através de experimentos entre migração e clonagem.
- Criação de modelo matemático baseado nesses experimentos:
  - simulação de uma rede de estações de trabalho pessoais homogêneas e das *tarefas* que são executadas nelas;
  - estudo de diferentes *graus de vivacidade* e de seus impactos.

2. Visão Geral do MobiGrid
2.1 Ambientes para agentes móveis

- MobiGrid [1] é um arcabouço de suporte a agentes móveis em grades baseado em Java.
- Por que Java?
  - Java é uma linguagem robusta, popular e moderna;
  - Comparação entre ambientes para agentes móveis [2]:
    - Grasshopper [3] e Aglets [4]: ambos em Java com notas 9.25 e 10.15;
    - Na época, Jade obteve nota 15.75.

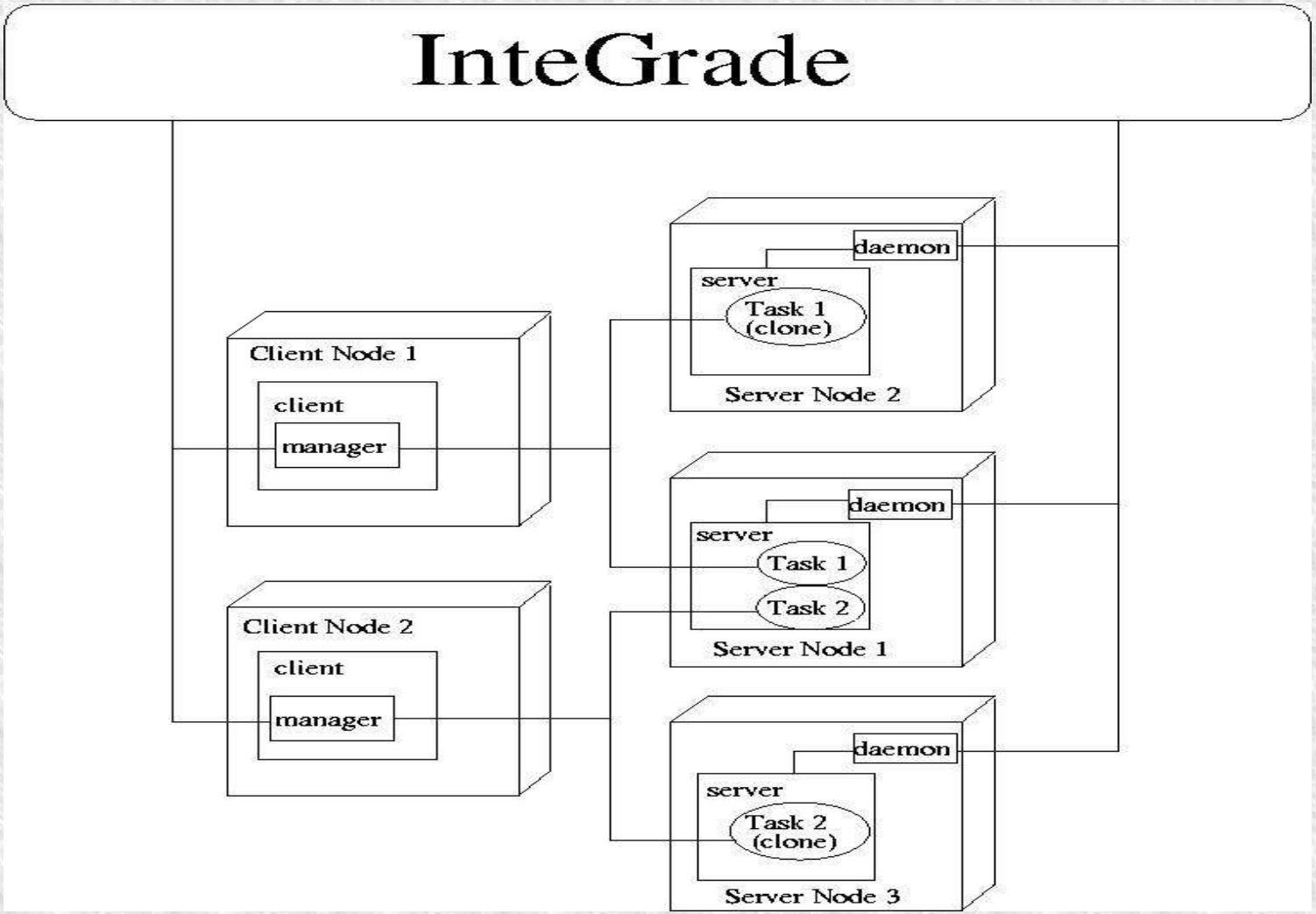
2. Visão Geral do MobiGrid
2.2 Arquitetura

- *tarefa:* aplicação com longo tempo de execução encapsulada dentro de um agente móvel.
- *gerente*: registra as *tarefas*. Deve estar ativo na máquina de quem submeteu as *tarefas*:
  - migração;
  - vivacidade.

2. Visão Geral do MobiGrid
2.2 Arquitetura (cont.)

- *servidor*: instalado em cada máquina que disponibiliza recursos para a infra-estrutura.
- *daemon*: verifica se a máquina local está ociosa ou não.
- *client*: responsável pela submissão de *tarefas* e hospedagem do *gerente*.

2. Visão Geral do MobiGrid
2.3 Diagrama da Arquitetura



3. Experimentos

- Para criarmos o modelo matemático, realizamos alguns experimentos que nos encaminharam para algumas suposições de simplificação.
- Testes realizados em um computador Athlon XP 2500 (1830 MHz) com 512 Mb de RAM, executando GNU/Linux 2.6.7.
- Foram desligadas as otimizações de migração do Aglets.

3. Experimentos
3.1 Clonagem e Migração

- 5 medições de tempo de migração e de clonagem.
- Primeiro experimento:
  - *tarefa* de ordenação de inteiros, operando sobre 100.000, 200.000 e 300.000 números.
- Segundo experimento:
  - *tarefa* que encapsula um algoritmo exponencial $O(m^n)$ trivial para resolução do problema do makespan.

3. Experimentos
3.1 Clonagem e Migração (cont.)

- Experimentos para a *tarefa* de ordenação:

	Migração – M	δ(M)	Clonagem	δ(C)	M/C
100000	33203,2	1786,83	135,6	1,34	244,86
200000	64156,4	2904,77	361,2	4,09	177,62
300000	92794,2	293,42	427,8	18,1	216,91

3. Experimentos
3.1 Clonagem e Migração (cont.)

- Experimentos para a *tarefa* do makespan:

	Migração – M	δ(M)	Clonagem – C	δ(C)	M/C
(2;26)	3698	135,03	15,8	3,49	234,05
(2;28)	3769,2	248,62	15,6	2,3	241,62
(2;30)	3671	181,17	15,6	2,7	235,32

3. Experimentos
3.2 Clonagem e Execução

- Mesma *tarefa* de ordenação já descrita operando sobre 100.000, 200.000 e 300.000 inteiros.
- 5 tomadas de tempo de execução da tarefa sozinha no servidor e executando com uma clonagem e migração desse clone.

3. Experimentos
3.2 Clonagem e Execução (cont.)

- Experimentos:

	Simples – S	δ(S)	Com migração	δ(M)	M/S
100000	18833,6	868,5	20729,8	529,56	1,1
200000	80637,8	2340,83	84219,8	2802,44	1,04
300000	204950,8	6643	204557,4	6161,34	1

3. Experimentos
3.3 Suposições de Simplificação

- O custo de clonagem de uma *tarefa* é zero:
  - poderia ainda ser computado junto com o tempo de migração;
  - clonagem depende apenas de operações rápidas em memória.
- A migração de uma *tarefa* clone não interfere com o tempo de execução da *tarefa* clonada.
  - migração executa operações que podem ser tratadas por DMA.

3. Experimentos
3.4 Restrições

- Uma *tarefa* por máquina:
  - garante as validade das simplificações;
  - é uma restrição válida, visto que com muitas tarefas em um *servidor*, há uma degradação de desempenho.
- Uma *tarefa* que está sendo clonada, durante seu tempo de clonagem, não pode ser clonada de novo.

4. Modelo Matemático

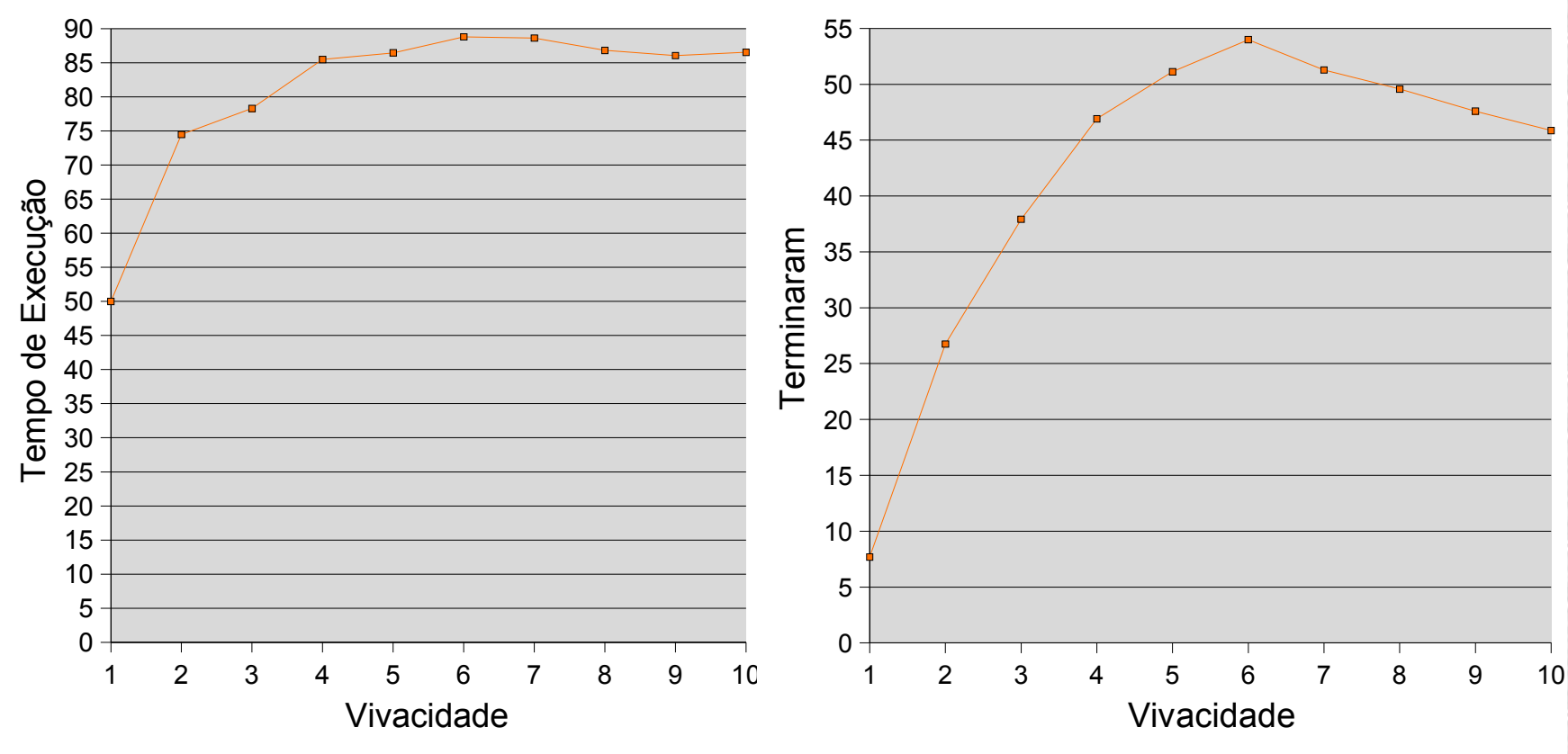
- Simulação probabilística discreta:
  - *tarefa*: descrita por um vetor $(x, y) \in Z^*_+ \times Z^*_+$.
  - *máquina*: representa o servidor. Cada máquina tem probabilidades $(p_1, p_2) \in S \times S$, onde $S = \{x: x \in R, 0 \leq x \leq 1\}$.
- Para nosso algoritmo, utilizaremos o conceito de *grupo de tarefas*, o qual representa as *tarefas gêmeas*.

```
G <- {grupos de tarefas}
D <- {} /* conjuntos dos grupos mortos  */
F <- {} /* conjuntos dos grupos que terminaram */
para todo e ϵ G faça
   aloque uma máquina para a primeira tarefa de e
num <- |G|; time <- 0;
enquanto (|D| + |F| < num) faça
   tempo++
   para toda máquina m faça
      desligue ou ligue m de acordo com p1 e p2
   para todo d ϵ G faça
      para cada tarefa t que não necessita ser clonada
         faça um passo de execução em t
      se todas tarefas t de d precisam ser clonadas
      então
         G <- G\{d}; D <- D U {d}
         libere as máquinas das tarefas de d
      senão se d tem uma tarefa que terminou execução
         G <- G\{d}; F <- F U {d}
         libere as máquinas das tarefas de d
      senão se tempo = 0 (mod b) então
         comece a clonagem e migração das tarefas
         cujas  máquinas estão mortas
      para cada tarefa t migrante
         faça um passo de migração ou recomece o
         processo
```

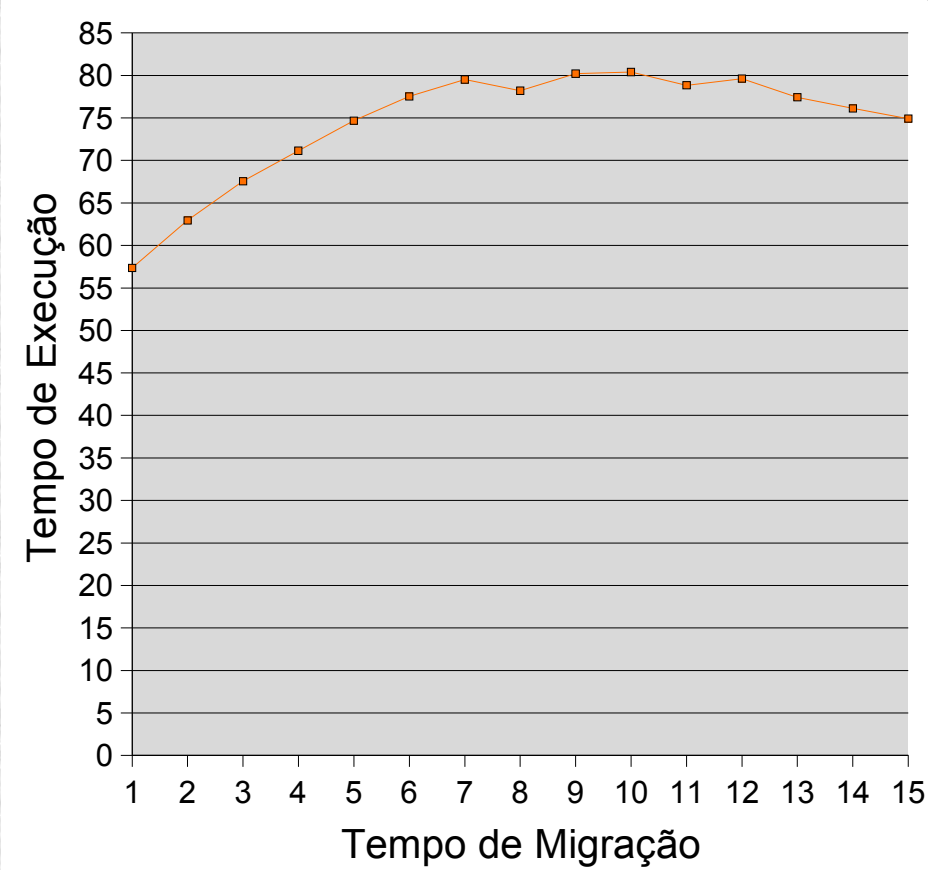
5. Experimentos

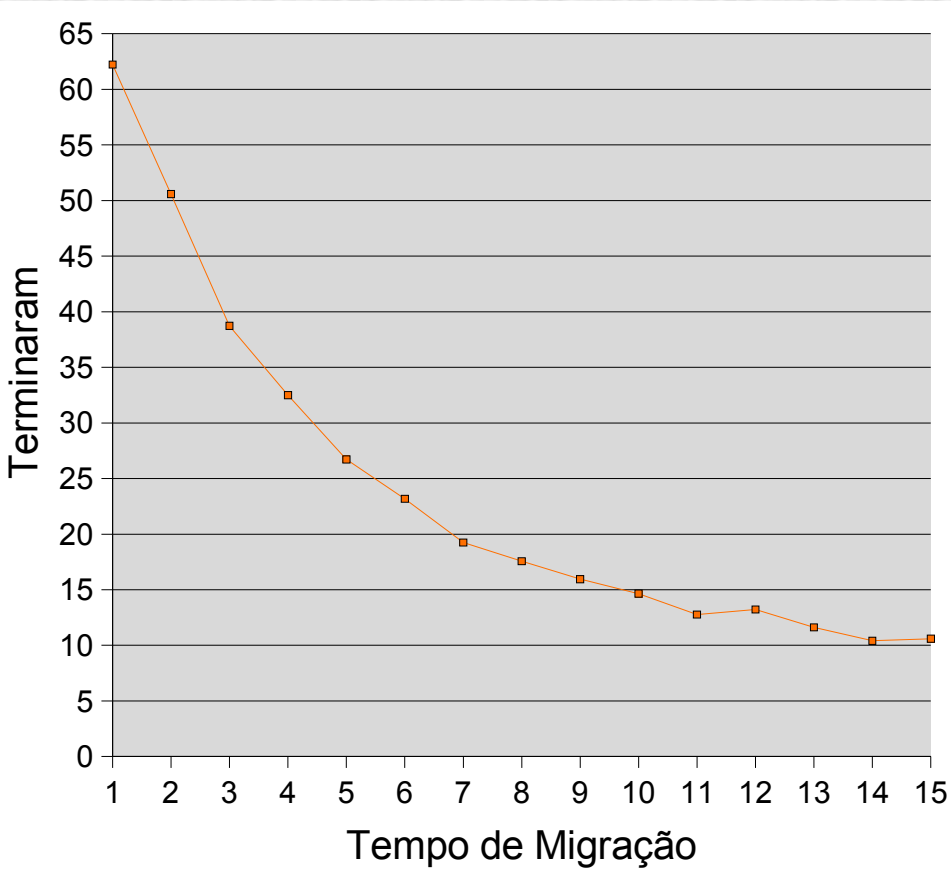
- 100 experimentos, cada um lançando 100 *grupos de tarefas*.
- Variação de um só parâmetro, com a fixação dos demais.
- Valores default:
  - TE = 50, TM = 5
  - p1 = 0,05, p2 = 0,1
  - l = 2, m = 500, n = 100

5. Experimentos
5.1 Parâmetros da *tarefa - l*

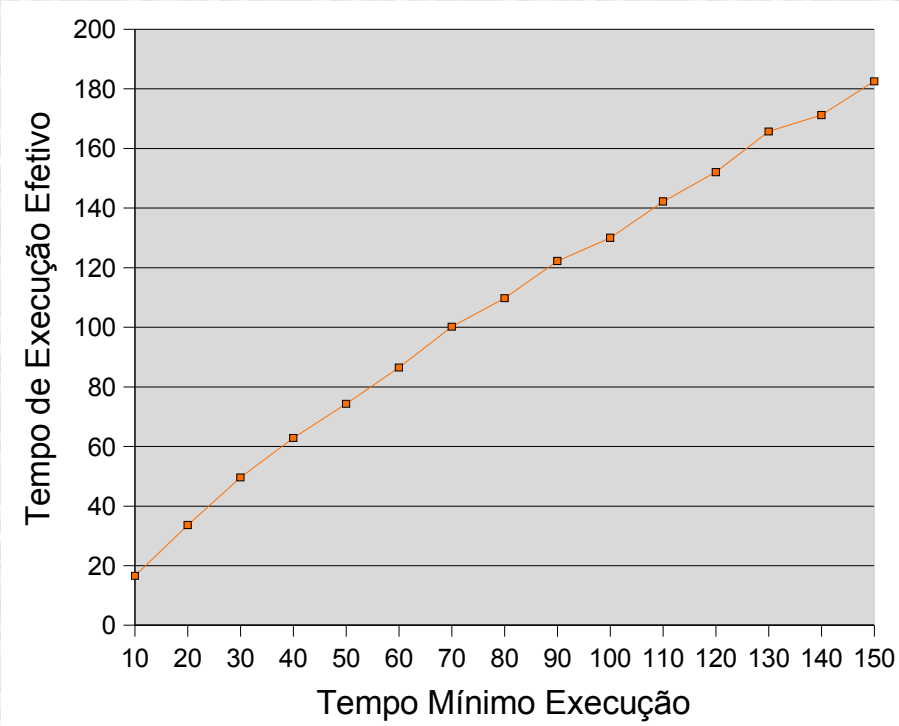


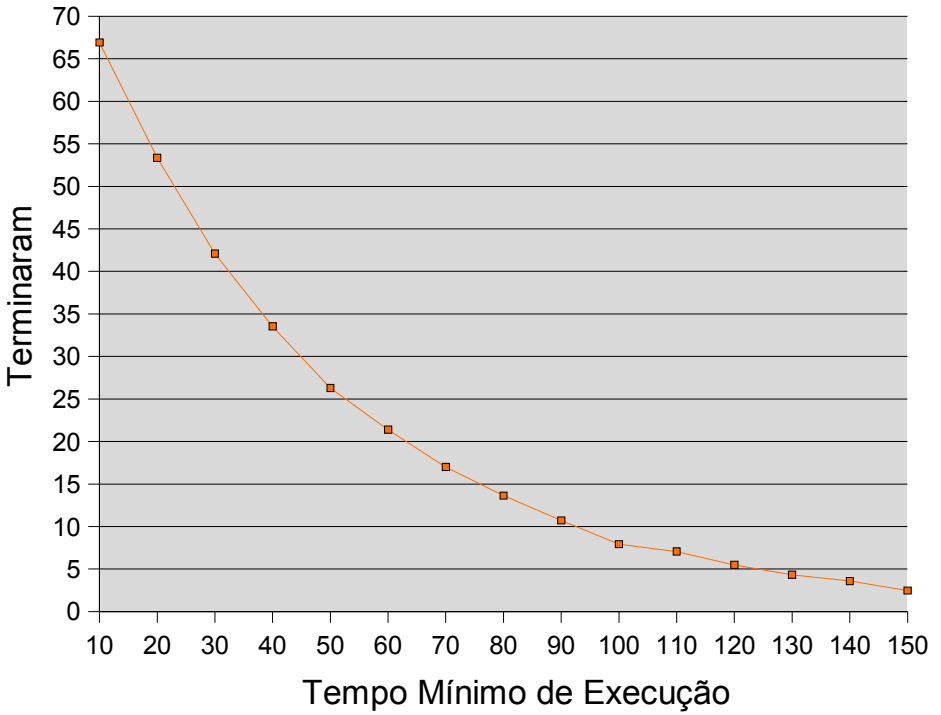
5. Experimentos
5.1 Parâmetros da *tarefa* - *TM*



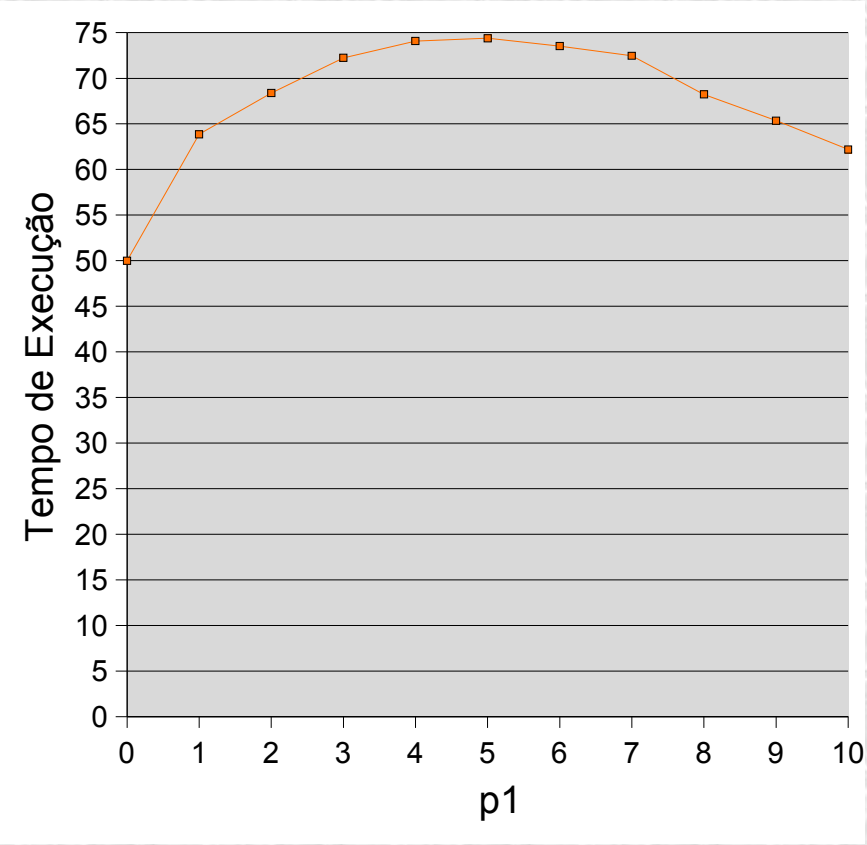


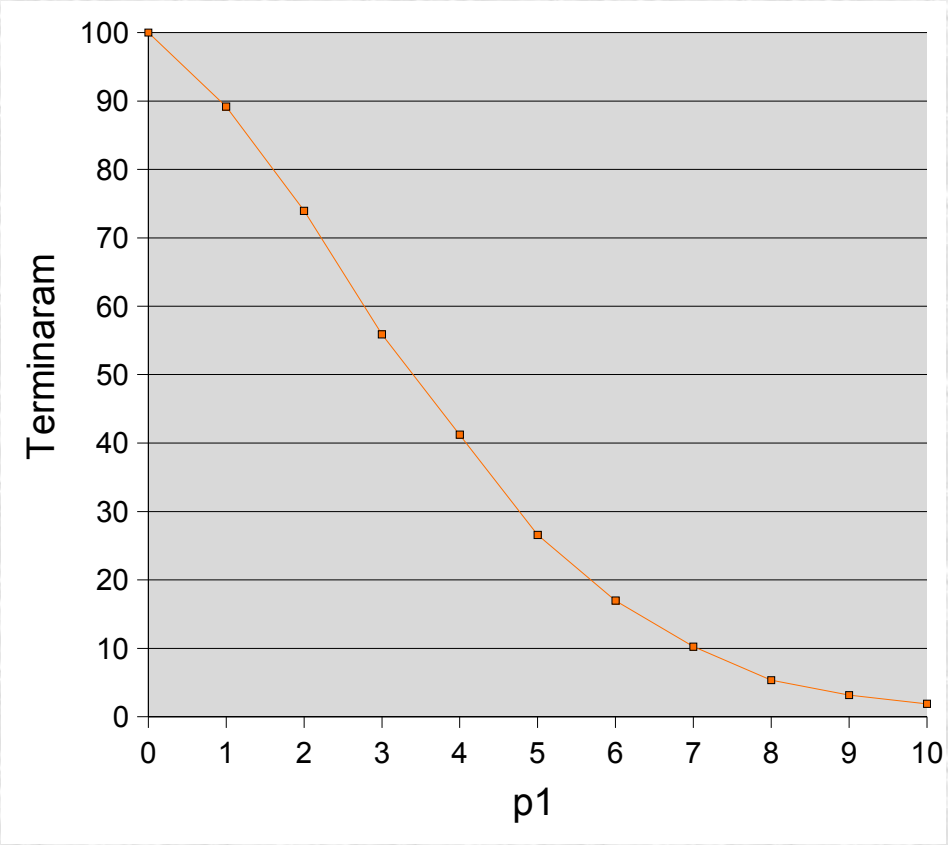
5. Experimentos
5.1 Parâmetros da *tarefa* - TE



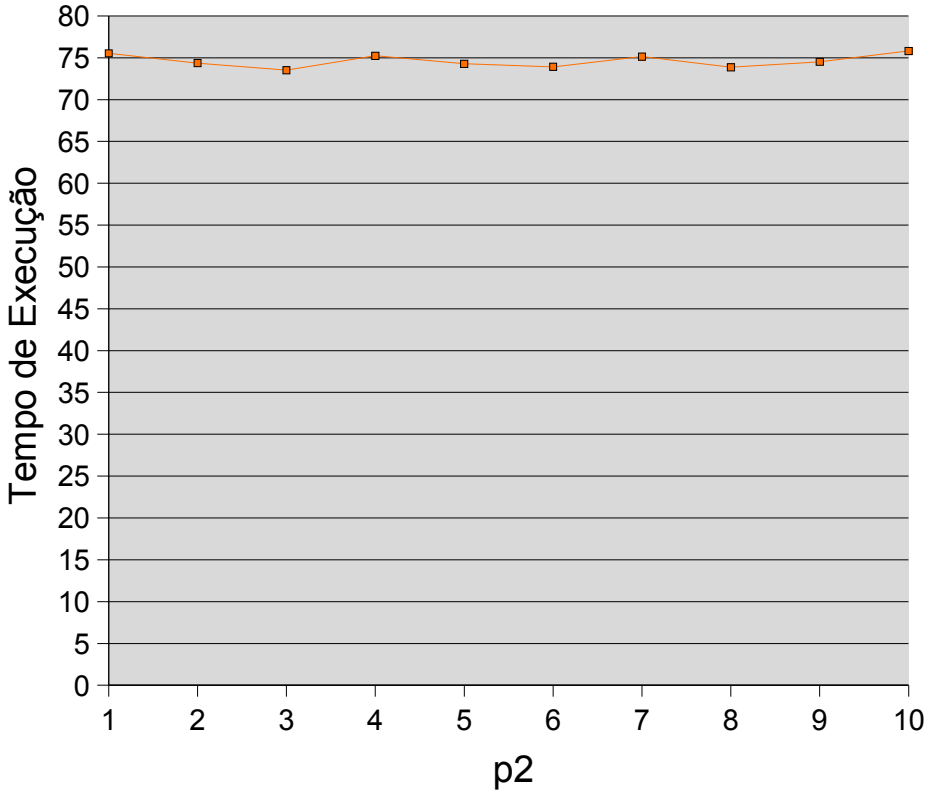


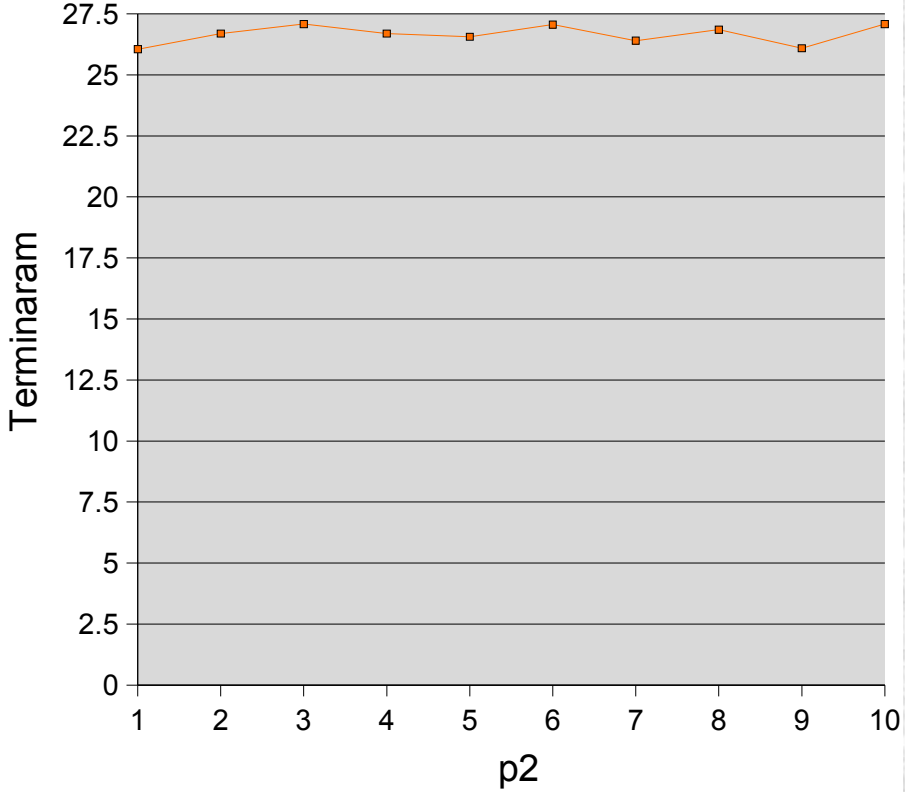
5. Experimentos
5.2 Parâmetros do ambiente - p1



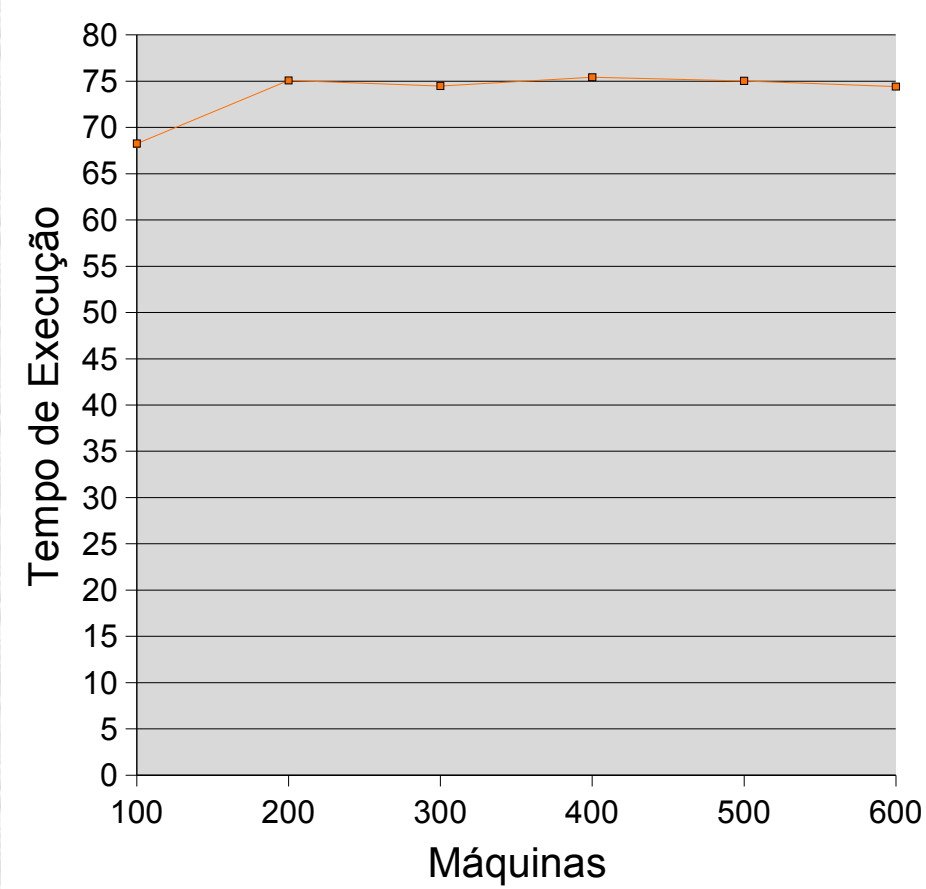


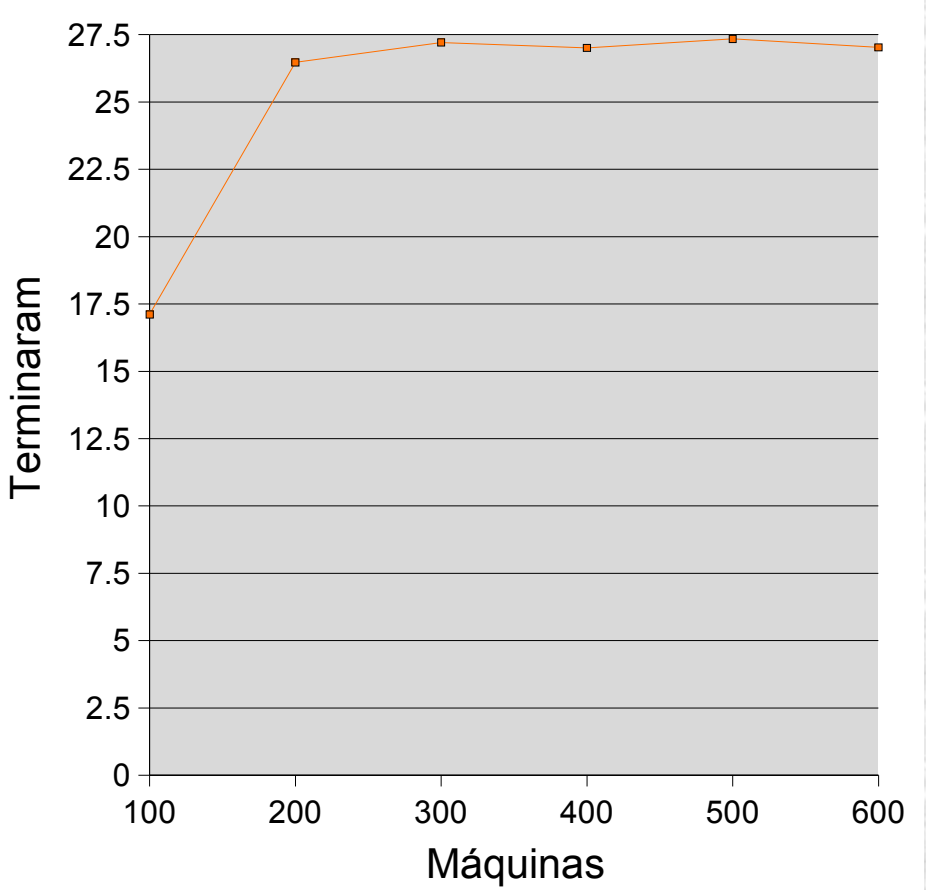
5. Experimentos
5.2 Parâmetros do ambiente - p2





5. Experimentos
5.2 Parâmetros do ambiente - m





6 - Referências

- [1] Rodrigo M. Barbosa e Alfredo Goldman. MobiGrid – Framework for Mobile Agents on Computer Grid Environments. In *Mobility Aware Technologies and Applications (MATA 2004)*, páginas 147-157. Springer-Verlag 2005.
- [2] Josef Altmann, Franz Gruber, Ludwig Klug, Wolfgang Stockner e Edgar Weppl. Using Mobile Agents in Real World: A Survey and Evaluation of Mobile Agents Platforms. In *Proceedings of the 2nd Workshop on Infrastructure for Agents, MAS, and Scalable MAS at Autonomous Agents*, 2001. http://www.umcs.maine.edu/~wagner/workshop/05_altmann_et_al.pdf
- [3] IKV++ GmbH. Grasshopper Basics and Concepts Release 2.2. Technical Report, IKV++ Technologies AG, março de 2001. http://www.grasshopper.de
- [4] Mitsuro Oshima e Guenter Karjoth. Aglets Specification 1.0. Technical Report, IBM, maio de 1997. http://www.research.ibm.com/trl/aglets/spec10.html