

Análise de padrões em grades computacionais oportunistas

Danilo M. R. Conde

11 de março de 2008

- 1 **Introdução**
 - Motivação
 - Objetivo
 - InteGrade
 - Escalonamento
- 2 **Análise de Agrupamentos**
 - Conceitos
 - Arcabouço
 - Algoritmo k -Médias
 - Eficácia no reconhecimento de padrões
- 3 **Implementação**
 - Coleta de Dados
 - Cálculo dos Padrões
 - Predição
- 4 **Experimentos**
 - Escalonamento
 - Sobrecarga
- 5 **Conclusões**

Sumário

- 1 **Introdução**
 - Motivação
 - Objetivo
 - InteGrade
 - Escalonamento
- 2 **Análise de Agrupamentos**
 - Conceitos
 - Arcabouço
 - Algoritmo k -Médias
 - Eficácia no reconhecimento de padrões
- 3 **Implementação**
 - Coleta de Dados
 - Cálculo dos Padrões
 - Predição
- 4 **Experimentos**
 - Escalonamento
 - Sobrecarga
- 5 **Conclusões**

Motivação

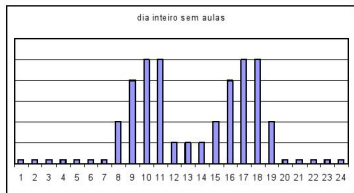
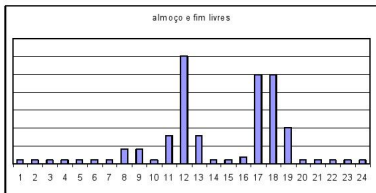
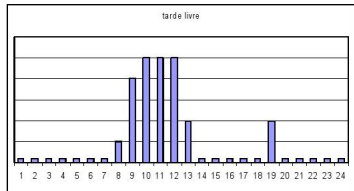
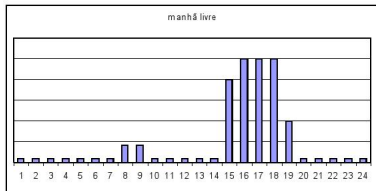
- Origem das grades
- Grande poder computacional com recursos de baixo custo
- Grades **oportunistas**
- Previsão de ociosidade

Objetivo

Desenvolver um sistema que seja capaz de prever a ociosidade de recursos (CPU, memória etc.) compartilhados com a grade.

- monitoração da utilização de recursos
- aprendizado de padrões de uso
- detecção do modo de utilização atual
- predição da utilização futura

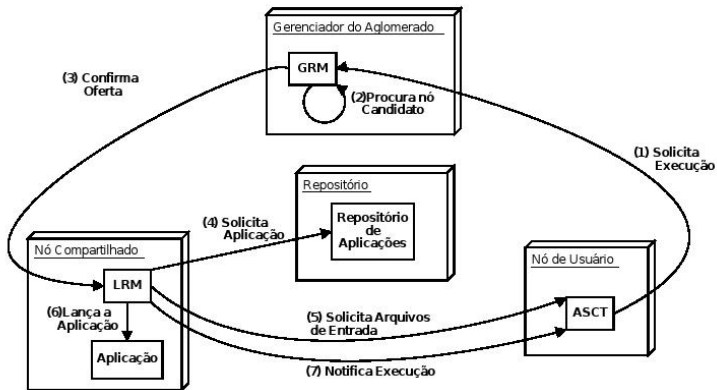
Padrões de uso



InteGrade

- Provê uma infra-estrutura para a implantação de grades oportunistas
- Desenvolvido por algumas universidades brasileiras: USP, UFG, UFMS, PUC-Rio e UFMA
- Diversas linhas de pesquisa

Ciclo de vida de uma aplicação no InteGrade



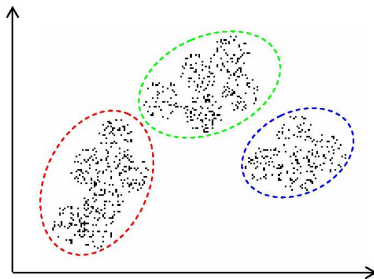
Escalonamento

- Diversos algoritmos na literatura
 - *bag of tasks*, algoritmos com barreiras de sincronização etc.
- Muitos utilizam algum tipo de previsão
 - tempo de execução da tarefa
 - capacidade de processamento disponível
- Yang, Schopf, Foster: predição homeostática ou baseada em tendência
- *Network Weather System*: seleção automática da melhor estratégia de predição

Sumário

- 1 Introdução
 - Motivação
 - Objetivo
 - InteGrade
 - Escalonamento
- 2 **Análise de Agrupamentos**
 - Conceitos
 - Arcabouço
 - Algoritmo k -Médias
 - Eficácia no reconhecimento de padrões
- 3 Implementação
 - Coleta de Dados
 - Cálculo dos Padrões
 - Predição
- 4 Experimentos
 - Escalonamento
 - Sobrecarga
- 5 Conclusões

- *clustering*, análise de conglomerados, *cluster analysis*
- É uma técnica de aprendizado não-supervisionado
- Separação de objetos em grupos de objetos semelhantes
- Não se trata de um técnica bem especificada, mas de vários métodos com o mesmo objetivo



Arcabouço - Anderberg

- 1 Escolha dos dados a serem analisados
- 2 Seleção das variáveis
- 3 Homogeneização das variáveis
- 4 Medidas de semelhança
- 5 Escolha dos algoritmos e implementação
- 6 Número de agrupamentos
- 7 Interpretação dos resultados

Algoritmo k -Médias

- Número k de agrupamentos é pré-definido
- Algoritmo:
 - 1 Escolha k objetos e crie uma partição com cada um deles
 - 2 Insira cada objeto que ainda não está em nenhuma partição à partição da qual ele está menos distante.
 - 3 Calcule a distância de cada objeto a cada uma das partições. Se a partição mais próxima não for a que ele está, mova-o para ela.
 - 4 Repita o passo 3 até convergir, ou seja, não existir mais nenhum objeto que precise trocar de partição

Dissertação de mestrado de Germano C. Bezerra:

- eficácia da análise de agrupamentos no reconhecimento de padrões de uso de recursos
- ociosidade de recursos é previsível

Outros resultados:

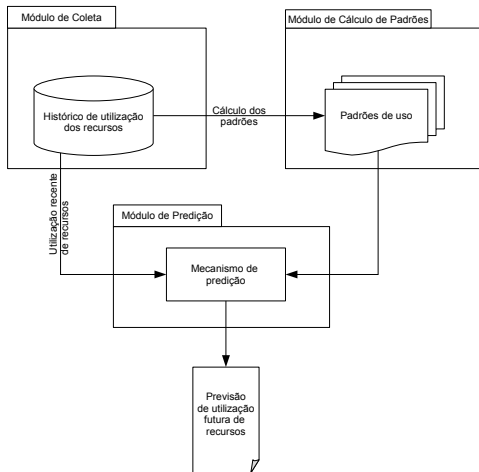
- variação da utilização é melhor que a própria utilização
- algoritmos: resultados semelhantes

Sumário

- 1 Introdução
 - Motivação
 - Objetivo
 - InteGrade
 - Escalonamento
- 2 Análise de Agrupamentos
 - Conceitos
 - Arcabouço
 - Algoritmo k -Médias
 - Eficácia no reconhecimento de padrões
- 3 Implementação**
 - Coleta de Dados
 - Cálculo dos Padrões
 - Predição
- 4 Experimentos
 - Escalonamento
 - Sobrecarga
- 5 Conclusões

- C++, CppUnit
- Independente dos outros módulos do InteGrade

Arquitetura interna do LUPA



Coleta

- CPU e memória
- 5 minutos
- arquivo de log
- estimativas
- dependente de S.O. (no Linux, os dados são obtidos do /proc)

Cálculo dos Padrões



UsageDatas são objetos que contêm as medidas de utilização num período de 48 horas.

Clusters são objetos que representam os padrões encontrados pelo algoritmo de análise de agrupamentos. Possuem um *objeto representativo*.

Processo de análise de agrupamentos

1 *Escolha dos dados a serem analisados*

UsageDatas que satisfazem as seguintes restrições:

- não tem mais que 6 meses de idade
- no máximo 10% dos dados são estimativas

2 *Seleção das variáveis*

- Padrões são calculados por recurso separadamente.
- Variação das medidas

3 *O que agrupar*

Objetos `Clusters`, que equivalem aos padrões de uso, e são representados por objetos representativos.

4 *Homogeneização*

Não houve homogeneização das variáveis.

Processo de análise de agrupamentos (*cont.*)

5 *Medidas de semelhança*

Distância euclideana:
$$d(a, b) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$

6 *Algoritmo*

Implementação própria do k -Médias.

Fase de inicialização: ordem da lista de entrada é aleatorizada

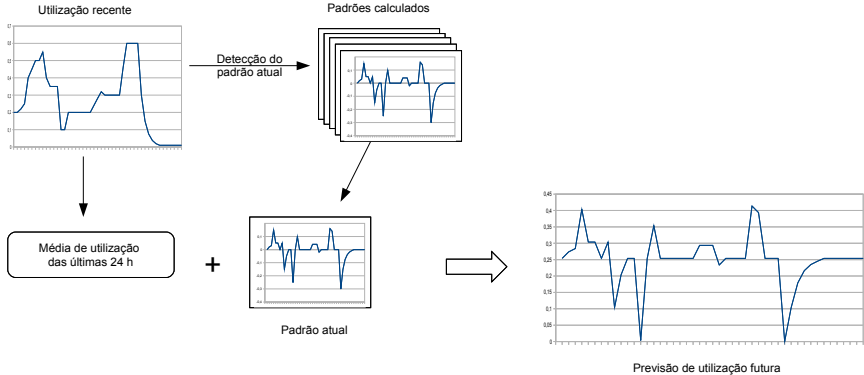
7 *Número de agrupamentos*

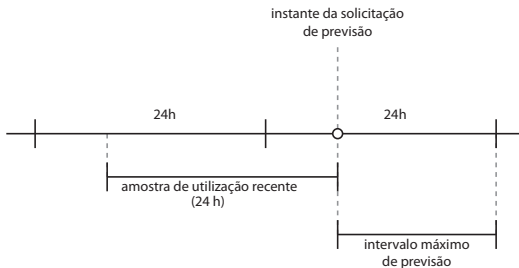
Cinco

8 *Interpretação dos resultados*

Implementação e Experimentos

Predição





```
1) double[] Lupa::getPrediction(resource r, int hours)
```

```
2) bool Lupa::canRunGridApplication(double freeCpuRequired,  
double freeMemoryRequired)
```

Recálculo dos padrões

Recálculo executado uma vez por dia.

Escalonamento

Algoritmo de escalonamento não foi alterado, no entanto, tornou-se mais eficaz devido as melhorias no módulo de previsão.

Sumário

- 1 **Introdução**
 - Motivação
 - Objetivo
 - InteGrade
 - Escalonamento
- 2 **Análise de Agrupamentos**
 - Conceitos
 - Arcabouço
 - Algoritmo k -Médias
 - Eficácia no reconhecimento de padrões
- 3 **Implementação**
 - Coleta de Dados
 - Cálculo dos Padrões
 - Predição
- 4 **Experimentos**
 - Escalonamento
 - Sobrecarga
- 5 **Conclusões**

Objetivo

Demonstrar que a previsão de utilização implementada pelo LUPA pode aumentar a eficácia do escalonamento de aplicações distribuídas

Dados: coletor de dados executado em 15 máquinas a partir de dezembro/2006

Ensaio

Comparação do desempenho de 4 algoritmos de escalonamento:

- round robin
- can run (50% de CPU e 100MB de memória disponíveis)
- get prediction
- last 4h

Ensaio (*cont.*)

Cada ensaio é uma seqüência de testes.

Parâmetros de um ensaio:

- n — quantidade de nós
- h — tempo de execução da aplicação
- m — quantidade mínima de máquinas disponíveis
- d — percentual máximo de disponibilidade de CPU média

$$\frac{\sum_{m \in E(t)} \text{utilizacao}(m, t, h)}{|E(t)|} < d$$

Um teste a cada hora dos dias elegíveis para execução de testes.

Ensaio (cont.)

Desempenho de cada algoritmo:

$$desempenho_i(t) = \frac{\sum_{m \in M_i(t)} utilizacao(m, t, h)}{n}$$

$$desempenho_i(t) \in [0, 1]$$

Ensaio (cont.)

Saída de cada teste:

- instante de teste;
- quantidade de máquinas elegíveis para esse instante de teste;
- desempenho do round robin (200 simulações);
- desempenho do can run (200 simulações);
- desempenho do get prediction;
- desempenho do last 4h ;
- desempenho do pior escalonamento;
- desempenho do melhor escalonamento.

Instante de teste	#	rrobin	can run	get pr.	last 4h	pior	melhor
2007-11-05 19:00	5	0.632	0.887	0.999	0.999	0.251	0.999
2007-11-08 01:00	5	0.691	0.992	0.999	0.995	0.241	0.999
2007-11-09 01:00	5	0.660	0.994	0.999	0.999	0.158	0.999

$n = 2$, $h = 2$, $m = 4$ e $d = 0.7$

Ensaio (*cont.*)

Visão sintética do resultado do ensaio:

	can run	get prediction	last 4h
round robin	83.3 - 16.7 - 0.0	83.3 - 16.0 - 0.7	71.3 - 16.0 - 12.7
can run		57.3 - 42.0 - 0.7	46.7 - 32.7 - 20.7
get prediction			4.0 - 72.7 - 23.3

$$n = 2, h = 2, m = 4 \text{ e } d = 0.7$$

$[i, j]$ é uma comparação entre o algoritmo da i -ésima linha com o algoritmo da j -ésima coluna: *pior-equivalente-melhor*

Ensaio (cont.)

Médias de ganho e perda (comparação com o round robin):

	can run	get prediction	last 4h
média de ganho	0.33	0.41	0.39
média de perda	nan	-0.18	-0.08

$$n = 2, h = 2, m = 4 \text{ e } d = 0.7$$

Análise dos resultados

Resumo comparativo:

	can run	get prediction	last 4h
desempenho	melhor que o round robin	melhor que o can run	semelhante ao get prediction
impacto do aumento de m/n	melhora no desempenho	melhora no desempenho	sem impacto
impacto do aumento de d	sem impacto	melhor desempenho com relação ao last 4h	pior desempenho com relação ao get prediction
impacto do aumento de h	sem impacto	sem impacto	sem impacto

Tempo de CPU

Arquivo	Dias válidos	Cálculo dos padrões	canRunGrid Application(...)	get Prediction(...)
jupiter.log	85	0.455333	0.000284314	0.000147059
plutao.log	95	0.394667	0.000280702	0.000144737
venus.log	120	0.447333	0.000295139	0.000145833
mercurio.log	199	0.870667	0.00239069	0.00147599
europa.log	103	0.457333	0.000299353	0.000141586
callisto.log	87	0.371333	0.000287356	0.000148467
saturno.log	116	0.451333	0.000301724	0.000150862
giga.log	74	0.362	0.000326577	0.000140766
motuca.log	112	0.485333	0.000282738	0.00014881

Memória

Arquivo de log	Dias válidos	Consumo (KB)
jupiter.log	85	2394,4
plutao.log	95	2518,4
venus.log	120	3016,8
mercurio.log	199	4754,8
europa.log	103	2763,2
callisto.log	87	2256,4
saturno.log	116	2964,8
giga.log	74	1874,8
motuca.log	112	2728,4

Sumário

- 1 Introdução
 - Motivação
 - Objetivo
 - InteGrade
 - Escalonamento
- 2 Análise de Agrupamentos
 - Conceitos
 - Arcabouço
 - Algoritmo k -Médias
 - Eficácia no reconhecimento de padrões
- 3 Implementação
 - Coleta de Dados
 - Cálculo dos Padrões
 - Predição
- 4 Experimentos
 - Escalonamento
 - Sobrecarga
- 5 Conclusões

Conclusões

- bons resultados nos experimentos, demonstrando ganhos significativos no desempenho de aplicações distribuídas no InteGrade
- qualidade do serviço
- potencial para ser pioneiro na previsão de utilização baseada em padrões de uso
- Limitações
 - apenas CPU e memória
 - implementação apenas para Linux

Dúvidas ?

Análise dos resultados

Relação m/n

- Ensaio: $n = 2$, $h = 4$, $m = 3$ e $d = 0.8$. Testes: 507.

	can run	get prediction	last 4h
round robin	78.3 - 17.8 - 3.9	88.8 - 9.5 - 1.8	80.7 - 9.5 - 9.9
can run		53.5 - 44.8 - 1.8	50.7 - 36.5 - 12.8
get prediction			7.7 - 77.1 - 15.2

- Ensaio: $n = 2$, $h = 4$, $m = 4$ e $d = 0.8$. Testes: 433.

	can run	get prediction	last 4h
round robin	88.0 - 9.0 - 3.0	93.5 - 5.8 - 0.7	83.1 - 5.5 - 11.3
can run		62.6 - 34.9 - 2.5	61.7 - 23.6 - 14.8
get prediction			9.2 - 73.4 - 17.3

- Ensaio: $n = 2$, $h = 4$, $m = 5$ e $d = 0.8$. Testes: 369.

	can run	get prediction	last 4h
round robin	89.4 - 10.6 - 0.0	92.7 - 6.8 - 0.5	80.2 - 6.5 - 13.3
can run		68.8 - 27.9 - 3.3	65.6 - 19.8 - 14.6
get prediction			10.6 - 69.1 - 20.3

- Ensaio: $n = 2$, $h = 4$, $m = 6$ e $d = 0.8$. Testes: 133.

	can run	get prediction	last 4h
round robin	92.5 - 7.5 - 0.0	97.7 - 0.0 - 2.3	81.2 - 0.0 - 18.8
can run		83.5 - 13.5 - 3.0	64.7 - 9.0 - 26.3
get prediction			9.0 - 54.1 - 36.8

Análise dos resultados (cont.)

Parâmetro d

- Ensaio: $n = 3$, $h = 6$, $m = 5$ e $d = 0.9$. Testes: 797.

	can run	get prediction	last 4h
round robin	87.0 - 12.3 - 0.8	90.6 - 9.0 - 0.4	89.6 - 10.0 - 0.4
can run		51.7 - 45.9 - 2.4	55.3 - 40.2 - 4.5
get prediction			8.0 - 83.7 - 8.3

	can run	get prediction	last 4h
média de ganho	0.17	0.21	0.20
média de perda	-0.03	-0.10	-0.07

- Ensaio: $n = 3$, $h = 6$, $m = 5$ e $d = 0.6$. Testes: 103.

	can run	get prediction	last 4h
round robin	73.8 - 26.2 - 0.0	87.4 - 12.6 - 0.0	87.4 - 12.6 - 0.0
can run		4.9 - 94.2 - 1.0	5.8 - 80.6 - 13.6
get prediction			1.0 - 83.5 - 15.5

	can run	get prediction	last 4h
média de ganho	0.36	0.38	0.33
média de perda	nan	nan	nan