Plataformas cooperativas: análise, incentivo e solução

Nicolas Bacic Lima, Daniel Cordeiro

[1]Escola de Artes, Ciências e Humanidades — Universidade de São Paulo
CEP 03828-000 – São Paulo – SP – Brasil

{nicolas.lima, daniel.cordeiro}@usp.br

Resumo. *A computação em grade é uma solução para organizações que desejam alcançar alto desempenho a custos mais baixos. Essa infraestrutura, entretanto, carrega uma maior complexidade se houver um conflito de interesses entre os participantes. Queremos que as organizações cooperem entre si ao mesmo tempo em que elas, individualmente, sejam beneficiadas. Esse problema, conhecido como MOSP, possui uma solução aproximada. Nesse trabalho iremos analisar o algoritmo que soluciona esse problema. Ao final, apresentamos nossas perspectivas para o estudo dessa solução para trabalhos futuros.*

1. Introdução

Até a década de 80, atividades intensivas de computação que demandavam um alto desempenho estavam associadas ao uso de supercomputadores [Fischborn 2006]. Tais máquinas eram extremamente caras e, portanto, sua posse estava restrita a algumas universidades privilegiadas, grandes corporações e poucos centros de pesquisa. Apesar do formidável avanço nas tecnologias de tais máquinas e também da queda em seu preço, tal situação perdura até os dias de hoje [Goldchleger 2004].

Organizações que demandam alto desempenho podem optar por um sistema de Computação em Grade (*Grid Computing* [Foster and Kasselman 2003]) que pode ser definido, de maneira bem abrangente, como uma infraestrutura de software capaz de interligar e gerenciar diversos recursos computacionais, possivelmente distribuídos por uma grande área geográfica, de maneira a oferecer ao usuário acesso transparente a tais recursos, independente da localização dos mesmos.

Essa solução, em contrapartida, requer cuidados para que o compartilhamento dos recursos seja feito de maneira justa. Um participante que compartilha seus recursos computacionais certamente irá querer, como contrapartida, que ele possa utilizar os recursos dos outros participantes. Um compartilhamento que não seja justo pode fazer com que os participantes simplesmente deixem de participar da plataforma, levando consigo seus recursos computacionais.

Queremos entender como incitar as organizações a compartilhar seus recursos e cooperarem entre si. Para isso precisamos considerar cada medida de desempenho que elas esperam otimizar, individualmente, ao trabalhar em conjunto com outras. Esse problema é conhecido na literatura como MOSP (problema de escalonamento multi-objetivo ou *Multi-Organization Scheduling Problem* [Cordeiro 2012]).

No contexto de computação paralela e distribuída, os recursos são os processadores disponíveis na plataforma e as atividades são as tarefas que devem ser executadas nesses processadores. Nesse artigo discutiremos um problema *offline*: toda informação para o escalonamento das tarefas é conhecida previamente pelo algoritmo.

Plataformas cooperativas, como as plataformas de computação em grade, são formadas pela união dos recursos de diversas organizações. Do ponto de vista macro, é razoável supor que o objetivo de desempenho seja utilizar as máquinas disponíveis (independente da organização) o melhor possível. Do ponto de vista de cada organização, seus usuários querem poder executar suas aplicações o mais rápido possível. Isso se traduz em dois objetivos, por vezes contraditórios: um objetivo global (otimizar o uso de todos os recursos) e objetivos locais às organizações (otimizar a execução das tarefas de uma organização).

Neste modelo, denotaremos por $O^{(k)}$ a k-ésima organização participante da plataforma. Cada organização possui $n^{(k)}$ tarefas $J^{(k)} = \{J_1^{(k)}, ..., J_n^{(k)}\}$ e dizemos que uma tarefa $J_i^{(k)} \in O^{(k)}$ termina de executar no instante de tempo $C_i^{(k)}$. Portanto, otimizar a execução das tarefas de $O^{(k)}$ significa minimizar o valor de seu *makespan* dado por $C_{max}^k = \max_{J_i^{(k)} \in O^{(k)}} \{C_i^{(k)}\}$. O objetivo global da plataforma é otimizar o ***makespan* global**, ou seja, $C_{max} = \max_k C_{max}^k$.

O problema MOSP consiste em escalonar as tarefas de todas as organizações afim de minimizar o *makespan* global com a seguinte restrição: o escalonamento resultante não pode fazer com que uma organização tenha sua função objetivo aumentada se comparado com o resultado que a organização obteria se escalonasse suas tarefas utilizando apenas suas próprias máquinas ($C_{max}^{(k)local}$). Mais formalmente, chamamos de MOSP o seguinte problema de otimização:

Minimizar C_{max} tal que, para todos os k $(1 < k \leq N)$

$$C_{max}^{(k)} \leq C_{max}^{(k)local}$$

O problema MOSP é sabidamente um problema NP-completo [Cohen et al. 2010]. Por isso, para obter-se soluções em tempo razoável é necessário utilizar-se de heurísticas para realizar o escalonamento. As heurísticas não produzem o melhor resultado possível (ótimo), mas pode-se provar que elas produzem resultados perto do ótimo.

O algoritmo de balanceamento de carga iterativo (ILBA [Cohen et al. 2010]) é uma heurística que redistribui a carga das organizações mais carregadas iterativamente. A ideia do algoritmo é de incrementalmente rebalancear a carga das organizações mais carregadas, sem que nenhuma tarefa atrase. Primeiro as organizações menos carregadas são rebalanceadas e então, uma por uma, cada organização é rebalanceada.

A heurística funciona da seguinte forma: primeiro, cada organização escalona suas próprias tarefas localmente e então as organizações são ordenadas de maneira não decrescente pelo *makespan*. Então, para k = 2 até N, tarefas da organização $O^{(k)}$ são reescalonadas sequencialmente e atribuídas para as organizações menos carregadas $O^{(1)}, O^{(2)}, ..., O^{(k)}$.

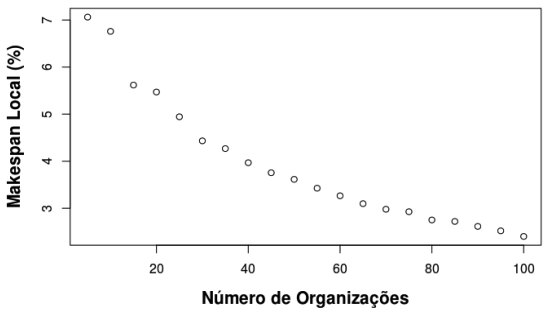
O objetivo desta pesquisa de Iniciação Científica é estudar o algoritmo ILBA e analisar a qualidade dos escalonamentos produzidos pelo algoritmo utilizando-se de simulações que caracterizem as tarefas encontradas em plataformas reais de computação de alto desempenho.

2. Experimentos

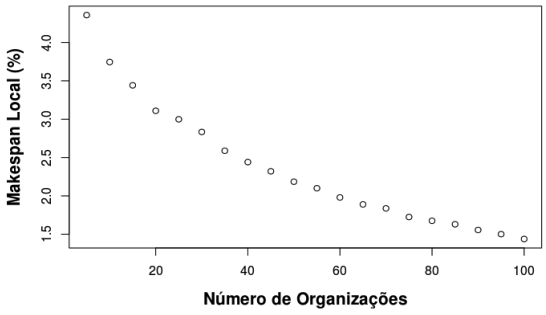
O algoritmo foi estudado a partir da descrição teórica apresentada em [Cohen et al. 2010] e implementada utilizando a linguagem de programação Go [Donovan and Kernighan 2015].

A versão mais atual do código desenvolvido nesse trabalho pode ser encontrado em `https://github.com/NickBacic/IC/` e permanece em contínuo desenvolvimento. Nessa sessão iremos apresentar o ganho global e local alcançado pelo algoritmo.

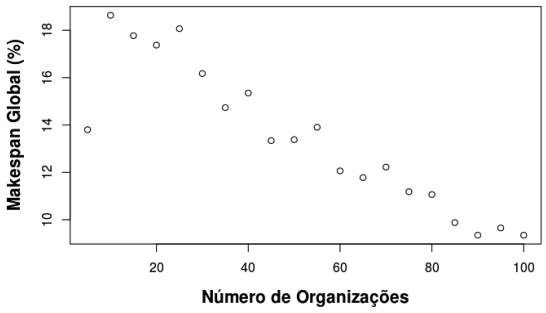
Nós conduzimos uma extensiva série de simulações para analisar os resultados obtidos pelo algoritmo. O *workload* foi aleatoriamente gerado com parâmetros que coincidem com o ambiente típico encontrado em plataformas de rede. Geramos, assim, um cenário de uma plataforma cooperativa com $(5, 10, 15, \ldots, 100)$ organizações. Para cada organização foi atribuído um número uniforme de tarefas (de 100 a 1000) com tamanhos escolhidos uniformemente ao acaso de 1 a 1000. Para cada combinação desses parâmetros foram geradas 100 instâncias aleatórias. Além de tudo, simulamos duas situações diferentes: inicialmente, o número de máquinas de cada organização era heterogêneo e aleatório (de 1 a 10); em seguida fixamos que cada organização possuía 5 máquinas. Consideramos que qualquer tarefa irá executar exatamente da mesma maneira em qualquer uma das máquinas disponíveis. Finalmente, os resultados foram coletados e podem ser conferidos nos gráficos abaixo.



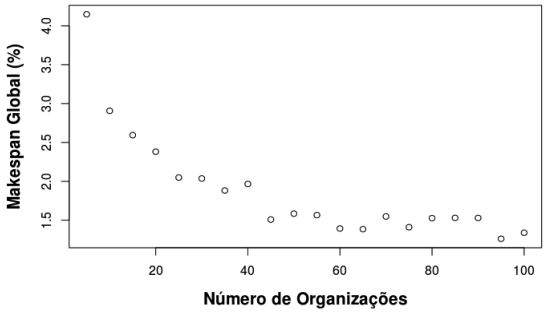
(a) Número de máquinas aleatório



(b) Número de máquinas fixo



(c) Número de máquinas aleatório



(d) Número de máquinas fixo

3. Análise

3.1. Makespan Local ($C_{max}^{(k)}$)

O primeiro item analisado empiricamente foi a melhora no desempenho percebida por cada organização, ou seja, o quanto $C_{max}^{(k)}$ melhorou em relação à $C_{max}^{(k)local}$. Em média há

um ganho de 3,92% na função objetivo local quando o número de máquinas é diferente e um ganho médio de 2,36% quando o número de máquinas é fixo.

O ganho local sempre é assegurado mesmo que ele não seja tão significativo. Isso se deve pois, nesse mecanismo, o escalonamento é feito da organização menos carregada até a mais carregada e portanto diminui o makespan da organização individualmente de uma maneira mais agressiva, minimizando a média local.

3.2. Makespan Global (C_{max})

O segundo item analisado empiricamente foi a melhora do ponto de vista macro, ou seja, o quanto C_{max} melhorou em relação à antes das organizações participarem da plataforma. Quando o número de máquinas é aleatório, em média, há um ganho de 13,45% mas quando o número de máquinas é fixo o ganho médio é de apenas 1,87%.

Daniel Cordeiro, em sua tese de doutorado [Cordeiro 2012], prova que o ILBA é um algoritmo de lista e que, portanto, os resultados obtidos pelo algoritmo são, no máximo, $2 - \frac{1}{m}$ vezes pior do que o melhor resultado possível (onde m é o número de recursos disponíveis).

4. Conclusão e perspectivas

Incitar a cooperação entre organizações não é um trabalho fácil. Mostramos que, apesar da dificuldade do problema apresentado, existe uma solução *satisfatória*. Além disso, estudamos qual o impacto de algumas variáveis sobre o resultado obtido pelo algoritmo.

O entendimento desse problema abre a possibilidade para novos trabalhos futuros como por exemplo, como encorajar a cooperação entre as organizações respeitando as condições do problema MOSP ao mesmo tempo que tentamos minimizar o gasto de energia.

Referências

Cohen, J., Cordeiro, D., Trystram, D., and Wagner, F. (2010). Analysis of multi-organization scheduling algorithms. In D'Ambra, P., Guarracino, M., and Talia, D., editors, *The 16th International Conference on Parallel Computing (Euro-Par)*, volume 6272 of *Lecture Notes in Computer Science*, pages 367–379. Springer.

Cordeiro, D. d. A. (2012). *The impact of cooperation on new high performance computing platforms*. PhD thesis, Université de Grenoble, Grenoble, France.

Donovan, A. A. A. and Kernighan, B. W. (2015). *The Go Programming Language*. Addison-Wesley Professional Computing Series. Pearson Education.

Fischborn, M. (2006). *Computação de alto desempenho aplicada a análise de dispositivos eletromagnéticos*. PhD thesis, Universidade Federal de Santa Catarina, Florianópolis, Brasil.

Foster, I. and Kasselman, C. (2003). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc.

Goldchleger, A. (2004). InteGrade: Um sistema de middleware para computação em grade oportunista. Master's thesis, Universidade de São Paulo.