Uma abordagem orientada a agrupamento de tarefas em balanceamento de carga distribuído

Vinicius M. C. T. de Freitas[1], **Laércio L. Pilla**[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brazil

vinicius.mct.freitas@gmail.com, laercio.pilla@ufsc.br

Resumo. *Estratégias de reescalonamento global garantem mais desempenho a aplicações de carga imprevisível ao manter uma distribuição eficiente de trabalho nas maiores plataformas paralelas. PackDrop é um algoritmo de escalonamento global distribuído que busca desempenho através do paralelismo em larga escala, enquanto diminui o sobrecusto de troca de mensagens levando o foco da tomada de decisão para o local.*

1. Introdução

Para simulações computacionais complexas de larga escala, como hidrodinâmica ou dinâmica molecular, ganho de desempenho é crucial para que o processamento seja concluído em tempo razoável. Mesmo as maiores plataformas paralelas sofrem ao tentar garantir esse ganho às aplicações, devido a problemas como custos de comunicação e desbalanceamento de carga [Pilla 2014]. Como as cargas dessas simulações podem ser instáveis, um escalonamento inicial satisfatório pode se mostrar inefetivo no futuro.

Estratégias de reescalonamento global podem ser usadas para mitigar este problema, movendo carga de trabalho para unidades de processamento (PUs) diferentes. Contudo, elas devem tomar pouco tempo para que não impactem a aplicação negativamente [Deveci et al. 2015].

Este trabalho apresenta **PackDrop**, uma estratégia de mapeamento global distribuído baseada em refinamento. Ela faz uso de informação local para criar e migrar grupos de tarefas, buscando reduzir tanto a comunicação quanto os custos de migração. Por sua natureza paralela, essa abordagem é capaz de aproveitar grandes sistemas evitando gargalos de comunicação e execução.

2. Metodologia

O processo de tomada de decisão em estratégias de escalonamento global pode ser abstraído para três grandes etapas, sendo estas: (i) dissipação de informação sobre o estado do sistema; (ii) tomada de decisão para o remapeamento de tarefas; e (iii) migração de tarefas.

Abordagens de escalonamento global distribuído previamente propostas avaliam cada tarefa individualmente para migração. Apesar de um balanceamento de carga mais preciso provido por esse tipo de abordagem [de Freitas and Pilla 2017], isso também pode levar a um aumento no tempo de decisão da estratégia.

Em **PackDrop** um conceito diferente é apresentado. Ao agrupar tarefas para migração, avaliando-as como grupo ao invés de individualmente, o foco da tomada de

decisão é local, reduzindo a comunicação nessa etapa da estratégia. Isso permite que menos informação seja compartilhada e uma tomada de decisão mais rápida seja realizada, ganhando desempenho nos passos (i) e (ii) do escalonamento.

O sistema paralelo utilizado neste trabalho foi o `Charm++`, que se mostrou ideal para a implementação do *PackDrop*. Ele é escalável e usado em diversas aplicações científicas, além de apresentar um *framework* de balanceamento de carga independente de aplicação [Acun et al. 2016], além de apresentar uma série de *benchmarks* e implementações de balanceadores de carga.

3. Estratégia Proposta: *PackDrop*

A Figura 1 mostra o fluxo de funcionamento do algoritmo principal, descrito a seguir:

Os dois primeiros passos (1), executados por todos os PUs, são realizar duas reduções, a primeira servindo para agregar a informação sobre a carga média e a segunda sobre o número de tarefas no sistema. Em seguida, cada um dos PUs se divide entre sobrecarregados e subcarregados (2).

Aqueles que são sobrecarregados irão começar a criar seus pacotes para transferência, enquanto os demais irão começar a propagar seus identificadores pela rede através de um protocolo *Gossip* [Menon and Kalé 2013], com o objetivo de informar os sobrecarregados a quem eles devem enviar suas tarefas (3).

Todos os processadores são sincronizados (4) para se iniciar o processo de tomada de decisão (5). A tomada de decisão é feita em conjunto pela fonte e destino de um pacote. Um PU sobrecarregado escolherá aleatoriamente uma série de PUs subcarregados a quem enviará pacotes. Os subcarregados devem calcular sua nova carga após o recebimento de um pacote e responder ao PU sobrecarregado, confirmando ou cancelando o remapeamento. Caso o envio seja cancelado, o PU sobrecarregado tentará enviar o pacote a outro subcarregado. Este processo se assemelha muito a um *three-way handshake* e garante que um PU subcarregado não receba pacotes demais, tornando-se o novo gargalo da aplicação paralela.

Esse processo descreve os passos (i) e (ii) apresentados na Seção 2, enquanto o passo (iii) é resolvido pelo *framework* do `Charm++`.

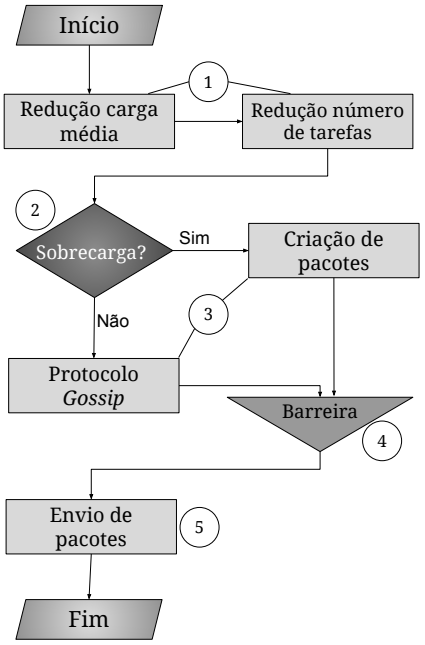

Figura 1. Fluxo do *PackDrop*, seguido por todos os PUs.

4. Avaliação de Desempenho

O principal ganho esperado com a abordagem aqui apresentada é **diminuição no volume de comunicação**. Isso acontece pois um PU é capaz de aceitar múltiplas migrações de uma vez, reduzindo o tempo de decisão.

Dentre os ***benchmarks*** utilizados para teste e avaliação, dois são perfis paralelos sintéticos e dois são *mini-apps* de aplicações de mundo real. Seus parâmetros estão discutidos na Tabela 1.

Benchmark	Dimensão	Comunicação
lb test	18990 (*med*); 36990 (*big*) tarefas	Anel
Stencil 3D	540 (bloco) e 12 (*array*)	Malha tridimensional
LULESH	140 (bloco) e 7 (*array*)	Malha tridimensional
LeanMD	$10 \times 15 \times 10$ (*med*); $15 \times 20 \times 15$ (*big*)	Muitos para muitos

Tabela 1. Parâmetros dos testes para avaliação de desempenho de cada um dos *Benchmarks*.

- **lb test**[1] : perfil paralelo para teste de balanceadores de carga.
- **Stencil 3D**: malha tridimensional que simula uma simples dissipação de calor.
- **LULESH**[2]: *mini-app* de simulações hidrodinâmicas. Segue o padrão computacional estêncil [Pereira et al. 2017].
- **LeanMD**[3]: simulação molecular baseado no potencial de *Lennard-Jones*.

4.1. Estratégias de Escalonamento

- **Greedy**: algoritmo de balanceamento centralizado guloso. Mapeia tarefas para núcleos independente do número de migrações.
- **Refine**: centralizado e baseado em refinamento, busca minimizar o número de migrações de tarefas, enviando-as de núcleos sobrecarregados para subcarregados.
- **Hybrid**: hierárquico, trabalha com diferentes algoritmos dependendo da hierarquia. Usa o *Greedy* dentro dos grupos e o *Refine* entre grupos.
- **Grapevine**: estratégia distribuída e baseada em refinamento. Distribuição probabilística de carga.

4.2. Ambiente de Testes

Todos os testes preliminares foram rodados em 16 nós do *cluster* **Genepi**[4], cada um contando com 2 processadores *Intel Xeon* E5420 QC @ 2.5GHz com 4 *cores* cada, totalizando **128 PUs**, 8GB de memória RAM DDR3 @ 1333MHz e 160GB de armazenamento (HDD). Os nós estão interconectados numa rede *InfiniBand* 20G. O sistema operacional utilizado foi o Ubuntu 14.04, a versão do `Charm++` adotada foi a 6.7.0 e o `GCC` foi utilizado na versão 5.4.0.

4.3. Resultados

A Figura 2 mostra os resultados em $speedup - 1$ relativos a execuções sem balanceamento de carga. Os resultados com ganhos tornam-se positivos e os demais negativos ou nulos.

Os *benchmarks* com padrões estêncil em malhas tridimensionais tiveram um impacto negativo do balanceamento de carga.

Balanceadores distribuídos destacam-se em comparação aos centralizados e hierárquicos conforme o tamanho do sistema cresce [Menon and Kalé 2013]. Portanto, apesar dos resultados semelhantes ao *Refine* em muitos casos, a tendência é que com o crescimento do sistema, a diferença entre eles também cresça.

[1]Disponibilizados com o `Charm++` através de: `http://charmplusplus.org/download/`

[2]Disponível em: `https://codesign.llnl.gov/lulesh.php`

[3]Disponível em: `http://charmplusplus.org/miniApps/`

[4]Experimentos apresentados neste trabalho foram realizados no Grid5000, mantido pelo grupo de propósitos científicos hospedado pela Inria e incluindo CNRS, RENATER e diversas Universidades e outras organizações (ver `https://www.grid5000.fr`).

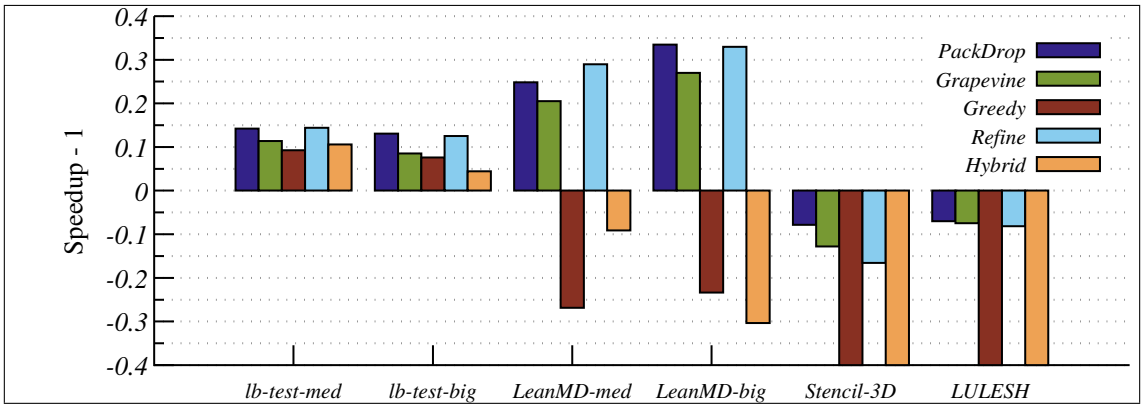

Figura 2. $Speedup - 1$ **de todos os *benchmarks* e aplicações testadas com cada estratégia de balanceamento de carga.**

5. Conclusão

Este trabalho apresentou *PackDrop*, uma estratégia de escalonamento global seguindo uma *abordagem orientada a pacotes*. O principal diferencial dela é a criação de pacotes assim como possibilita a migração de múltiplas tarefas de uma única vez.

Essa nova abordagem mostrou resultados promissores, tendo ganho de desempenho em todos os casos de teste experimentados em relação a outra estratégia distribuída observada, o *Grapevine*.

5.1. Trabalhos Futuros

É importante levar o *PackDrop* a plataformas de mais larga escala, uma vez que os testes realizados neste trabalho ainda não tem a dimensão necessária para mostrar o potencial das estratégias distribuídas.

Referências

Acun, B., Langer, A., Meneses, E., Menon, H., Sarood, O., Totoni, E., and Kalé, L. V. (2016). Power, reliability, and performance: One system to rule them all. *Computer*, 49(10):30–37.

de Freitas, V. M. C. T. and Pilla, L. L. (2017). Comparando diferentes ordenações de tarefas em balanceamento de carga distribuído. In *Anais da Escola Regional de Alto Desempenho do Rio Grande do Sul 2017*.

Deveci, M., Kaya, K., Uçar, B., and Catalyurek, U. V. (2015). Fast and high quality topology-aware task mapping. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 197–206.

Menon, H. and Kalé, L. (2013). A distributed dynamic load balancer for iterative applications. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 15:1–15:11, New York, NY, USA. ACM.

Pereira, A. D., Castro, M., Dantas, M. A. R., Rocha, R. C. O., and Góes, L. F. W. (2017). Extending openacc for efficient stencil code generation and execution by skeleton frameworks. In *2017 International Conference on High Performance Computing Simulation (HPCS)*, pages 719–726.

Pilla, L. L. (2014). *Topology-Aware Load Balancing for Performance Portability over Parallel High Performance Systems*. PhD thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do Sul.