Gmail **Rodolfo Pereira Araujo <rodoufu@gmail.com>**

Fwd: problema de otimização em dataflow

Igor Machado <igor.machado@gmail.com> 21 de outubro de 2017 11:25
Para: Rodolfo Pereira Araujo <rodoufu@gmail.com>

---------- Mensagem encaminhada ----------
De: **Igor Machado** <igor.machado@gmail.com>
Data: 8 de maio de 2017 17:28
Assunto: Re: problema de otimização em dataflow
Para: Alexandre Sena <asena@ime.uerj.br>, Paulo Gimenez <pjagimenez@gmail.com>

https://mathema.tician.de/dl/main.pdf

Em 8 de maio de 2017 17:23, Igor Machado <igor.machado@gmail.com> escreveu:

> Vou criar um problema toy que equivale aparentemente ao que eu preciso, então você me diz o que parece.
> Suponha que três operações Op1, Op2, Op3 deverão ser feitas de forma a reduzir um valor de entrada.
>
> ```
> int PROBLEM=1234; // Op1, Op2 e Op3 tem acesso a um único problema (poderia ser um vetor, algo mais complicado, aqui só um int serve)
> int Op1(int input) {
>     srand(PROBLEM + 100000 + input); // 100000 é constante única da Op1
>     sleep(100); // 100 miliseconds to process
>     return input+rand()%10 - 1; // 10% de chance de reduzir o valor de entrada
> }
> int Op2(int input) {
>     srand(PROBLEM + 200000 + input); // 200000 é constante única da Op2
>     sleep(200); // 200 miliseconds to process
>     return input+rand()%10 - 3; // 30% de chance de reduzir o valor de entrada
> }
> int Op3(int input) {
>     srand(PROBLEM + 300000 + input); // 300000 é constante única da Op3
>     sleep(500); // 500 miliseconds to process
>     return input+rand()%10 - 4; // 40% de chance de reduzir o valor de entrada
> }
> ```
>
> A regra é a seguinte, dado um valor de entrada 999999 em um nó principal, ele repassa esse valor aos nós Op1(999999), Op2(999999), Op3(999999). Assim que algum nó termina, eles retornam em um loop ao nó principal com o valor retornado. Se o valor retornado é menor, digamos 999995, repassamos novamente a todos os nós Op1(999995), Op2(999995), Op3(999995). Aquele(s) que já estava(m) livre(s) começa(m) imediatamente com o novo valor, quem está processando não é paralisado, pois pode retornar um valor possivelmente menor. O algoritmo acaba quando ninguém consegue mais reduzir o valor de entrada.
> Meu questionamento empírico é: é possível manipular a fila de espera dos dados de forma que esse comportamento seja obtido com dataflow? na otimização, alguns processos vão demorar mesmo mais do que outros, e às vezes três ou quatro melhorias podem ser feitas enquanto ele só processa uma, e o mais lógico seria ele processar a última enviada (com menor valor), pois é a ideia de uma busca local, buscando melhoras rapidamente, não uma metaheurística que diversifica tentativas.
>
> Em 8 de maio de 2017 17:22, Igor Machado <igor.machado@gmail.com> escreveu:
>
> > https://github.com/tiagoaoa/Sucuri/blob/master/examples/optimizaiton.py
> >
> > Em 3 de abril de 2017 20:53, Tiago Alves <tiago@ime.uerj.br> escreveu:
> >
> > > Pode botar as funções em c++ mesmo que a gente faz os wrappers pra python.
> > >
> > > -------- Original message --------

From: Tiago Alves <tiago@ime.uerj.br>
Date: 4/3/17 15:51 (GMT-03:00)
To: Igor Machado <igor.machado@gmail.com>
Cc: felipe França <felipe@cos.ufrj.br>, Leandro Marzulo <leandro@ime.uerj.br>
Subject: Re: problema de otimização em dataflow

Sim. Vou implementar exatamentr aquele exemplo para facilitar seu entendimento então.

Viajo hoje pra amherst, faço la na umass e te mando.

-------- Original message --------
From: Igor Machado <igor.machado@gmail.com>
Date: 4/3/17 15:34 (GMT-03:00)
To: "Tiago A.O.A." <tiago@ime.uerj.br>
Cc: felipe França <felipe@cos.ufrj.br>, Leandro Marzulo <leandro@ime.uerj.br>
Subject: Re: problema de otimização em dataflow

Blz Tiago, me enrolei um pouco agora pq tudo que tenho tá em C ou C++, então tô fatiando aqui as partes úteis pra poder migrar. De forma geral, aquele exemplo que eu tinha proposto é possível de executar na sucuri com esse flipflop?
Vou dar uma olhada no seu repositório e tentar executar aqui localmente.

Abraço,
Igor

Em 31 de mar de 2017 4:30 PM, "Tiago A.O.A." <tiago@ime.uerj.br> escreveu:
> Exemplo em: https://github.com/tiagoaoa/Sucuri/blob/master/examples/optimizaiton.py
>
> Acho que está bem intuitivo, está tudo no próprio exemplo.
>
> A única coisa que botei na Sucuri em si foi uma classe de node chamada FlipFlop para auxiliar. Se a função atribuída ao nó flipflop retornar False, o nó não envia nada pra ninguém. Se a função identifica que a solução é a melhor encontrada até o momento, propaga.
>
> Depois vou fazer um exemplo de algoritmo que tenha convergência com ponto flutuante e aí o tamanho do erro ficaria nessa função.
>
> Em 29 de março de 2017 19:38, Tiago A.O.A. <tiago@ime.uerj.br> escreveu:
>> Tem as heurísticas de otimização escritas em python para um primeiro teste?
>> A solução está pronta.
>>
>> On Mar 19, 2017 19:53, "Igor Machado" <igor.machado@gmail.com> wrote:
>>> Ótimo Tiago, nesse meio-tempo já preparo as funções corretas pra encaixar no mesmo framework.
>>> Agora, um detalhe importante que eu abstraí: a estrutura de dados PROBLEM (que era um int), vai acabar sendo um conjunto de matrizes e vetores armazenado tanto nas CPUs quanto GPUs... e o input passado por cópia seria na verdade um vetor com dados da solução na CPU e GPU.
>>> Sei que a GPU pode trazer alguns problemas, mas se houver uma função de incialização para cada worker, o dado seria passado via CPU e o worker faria sua cópia para GPU, guardando o ponteiro internamente para uso posterior. A única perda nesse caso seria caso dois workers estiverem ligados a uma mesma GPU, não precisaria de duas ou mais cópias da mesma estrutura de problema.
>>> Já o input pode ser passado normalmente via CPU, que internamente o worker faz a transferencia pra GPU, trabalha, libera a memória e retorna o resultado.
>>>
>>> Sobre seu próximo email, percebo que o grafo dataflow não tem relação com o problema abordado, por isso tentei reduzir a esse caso hiper simplório que só precisa de inteiros :)
>>>
>>> Em 19 de março de 2017 19:43, Tiago A.O.A. <tiago@ime.uerj.br> escreveu:
>>>> Sim. Na realidade já daria pra fazer isso na Sucuri ou na Trebuchet apenas com alterações para sincronização no código do usuário (programador). Mas vou incluir uma funcionalidade na sucuri para deixar isso transparente.
>>>>
>>>> Durante a semana eu faço isso e mostro como fica com esse benchmark artificial que você mandou.
>>>>
>>>> Em 19 de março de 2017 17:48, Igor Machado <igor.machado@gmail.com> escreveu:
>>>>> Pois é, como eu falei, entendo pouco de dataflow e parece que não se encaixa ao "dataflow tradicional".

Vou criar um problema toy que equivale aparentemente ao que eu preciso, então você me diz o que parece. Suponha que três operações Op1, Op2, Op3 deverão ser feitas de forma a reduzir um valor de entrada.

```
int PROBLEM=1234; // Op1, Op2 e Op3 tem acesso a um único problema (poderia ser um vetor, algo mais complicado, aqui só um int serve)
int Op1(int input) {
   srand(PROBLEM + 100000 + input); // 100000 é constante única da Op1
   sleep(100); // 100 miliseconds to process
   return input+rand()%10 - 1; // 10% de chance de reduzir o valor de entrada
}
int Op2(int input) {
   srand(PROBLEM + 200000 + input); // 200000 é constante única da Op2
   sleep(200); // 200 miliseconds to process
   return input+rand()%10 - 3; // 30% de chance de reduzir o valor de entrada
}
int Op3(int input) {
   srand(PROBLEM + 300000 + input); // 300000 é constante única da Op3
   sleep(500); // 500 miliseconds to process
   return input+rand()%10 - 4; // 40% de chance de reduzir o valor de entrada
}
```

A regra é a seguinte, dado um valor de entrada 999999 em um nó principal, ele repassa esse valor aos nós Op1(999999), Op2(999999), Op3(999999). Assim que algum nó termina, eles retornam em um loop ao nó principal com o valor retornado. Se o valor retornado é menor, digamos 999995, repassamos novamente a todos os nós Op1(999995), Op2(999995), Op3(999995). Aquele(s) que já estava(m) livre(s) começa(m) imediatamente com o novo valor, quem está processando não é paralisado, pois pode retornar um valor possivelmente menor. O algoritmo acaba quando ninguém consegue mais reduzir o valor de entrada.
Meu questionamento empírico é: é possível manipular a fila de espera dos dados de forma que esse comportamento seja obtido com dataflow? na otimização, alguns processos vão demorar mesmo mais do que outros, e às vezes três ou quatro melhorias podem ser feitas enquanto ele só processa uma, e o mais lógico seria ele processar a última enviada (com menor valor), pois é a ideia de uma busca local, buscando melhoras rapidamente, não uma metaheurística que diversifica tentativas.

Se você conseguir implementar um exemplo simples como esse, passar ele pro meu algoritmo vai ser facil.

Abraço,
Igor

Em 19 de março de 2017 16:12, Tiago Alves <tiago@ime.uerj.br> escreveu:

> Realmente me confundi ao comparar com o A*, que faz sua própria poda pela heurística. Mas mesmo assim, acho que uma visão global faria algumas podas mais cedo, estou certo?
>
> Quanto à Sucuri, sim funciona. Só preciso adicionar uma funcionalidade.
>
> O problema que vocês estão vendo é por confundir substrato com o sistema distribuído em si. Dataflow suporta qualquer algoritmo, você só precisa descrever o algoritmo em um grafo da forma correta. Um nó da árvore do seu algoritmo não deve ser um nó do grado dataflow. Os passos do algoritmo é que são nós do grafo.
>
> Resumo, o grafo dataflow tem tamanho fixo, não deve crescer.
>
> Sent from my Samsung Galaxy smartphone.
>
> -------- Original message --------
> From: Igor Machado <igor.machado@gmail.com>
> Date: 3/19/17 15:00 (GMT-03:00)
> To: Tiago Alves <tiago@ime.uerj.br>
> Subject: Re: problema de otimização em dataflow

Nunca tinha pensado em termos do A*, mas pode ser algo do tipo, embora o A* tenha uma função heurística que de fato garante um bom caminho de a estimativa de custo for subestimada. No meu caso, nada é garantido.
Então, implementar um algoritmo distribuído, com MPI, é totalmente possível, mas exatamente o que eu não queria fazer. Se eu faço ele distribuído, o código vira uma zona e nunca mais aproveito pra nada, e se fosse na Sucuri ou outro framework, seria totalmente aproveitável.
Outro ponto é que aparentemente isso é uma aplicação MISD (Multiple Instruction Single Data), o Marzulo percebeu isso, e é curioso que a gente nunca tinha visto outra aplicação dessa natureza.
Então fica a pergunta, é possível a Sucuri/dataflow suportar aplicações MISD?

Abraço,
Igor

Em 19 de março de 2017 14:06, Tiago Alves <tiago@ime.uerj.br> escreveu:

> Igor,
> entendi. É uma poda em um problema de busca.
>
> Só pra verificar: seria equivalente a resolver um A* em paralelo, mas informando aos workers o mais rápido possível que eles estão seguindo um caminho ruim?
>
> Essa solução de uma fila só não me parece muito interessante. Melhor elaborar um algoritmo distribuído.
>
> Aguarde um pouco que vou pensar.
>
> -------- Original message --------
> From: Igor Machado <igor.machado@gmail.com>
> Date: 3/18/17 23:25 (GMT-03:00)
> To: Tiago Alves <tiago@ime.uerj.br>
> Subject: Re: problema de otimização em dataflow
>
> Opa, ainda não tenho um código pronto Tiago (do DVND), somente as cinco operações CUDA prontas para cada busca Multi-Improvement (que eu apresentei no evento).
> Eu trabalho com um colega que chegou a implementar uma prévia do DVND multi-threading (sem MPI), mas ficou extremamente confusa, complexa e certamente não vai ajudar. Então ao invés de reprogramar usando MPI, pensei que poderíamos usar a Sucuri pra fazer essa tarefa, mas somente se isso se encaixar no dataflow de alguma forma!!! Não quero destruir o cerne do framework de vocês, se não fizer sentido o que eu preciso me diga por favor, mas eu tenho esperança de que se encaixe de alguma forma pq seria bom pra nós todos.
>
> Posso te explicar a ideia? O pseudocodigo do DVND tá no paper em anexo, mas talvez a ideia seja mais simples de entender.
>
> Ok, suponha que temos um nó principal que recebe uma solução do problema (permutação) juntamente com seu valor (suponha 500). Agora temos 5 nós que fazem buscas locais (operações) diferentes (swap, 2-opt, or1, or2 e or3). Minha versão multi-threading tinha 2 GPUs no PC, e 5 threads trabalhadoras (uma pra cada operação, mas eu alocava 2 em uma GPU e 3 na outra). Quando um nó retorna sua computação (suponha swap), ele retorna ao nó principal e apresenta seu resultado (suponha solução 400, numa minimização). A solução 400 seria guardada e a 500 descartada, e o nó swap recomeça com a solução 400. Agora suponha que o nó 2-opt termina com solução 450. Isso era melhor que 500, mas não ajuda agora que já temos 400, então a solução 450 é descartada e ele recomeça da solução 400. Agora o nó swap termina denovo, mas com solução 430 (e ele já partiu da 400), então não há mais o que fazer, e o swap fica paralizado até que tenha uma solução nova que compense. Agora o or-1 termina com solução 300 (tinha partido de 500), ele atualiza a melhor no nó principal para 300, e or-1 e swap recomeçam de 300 (swap estava paralisada e volta). O método termina quando todos os nós estão paralisados, ou seja, quando não há nova solução boa na fila de espera.
>
> Conversei muito com o Marzulo e tinhamos pensado que se a fila só guardasse a última informação relevante (de menor valor), ao invés de todas (devido ao loop no grafo), o problema estaria resolvido.

Algum insight de como resolver?

Abraço,
Igor

Em 18 de março de 2017 20:44, Tiago Alves <tiago@ime.uerj.br> escreveu:

> Igor,
> me manda um texto e um codigo, se ja tiver, do método de otimização que você quer paralelizar.
>
> Pelo que o marzulo falou eu acho que dou um jeito nisso em dois tempos e já disparamos nosso primeiro paper juntos.
>
> Abraço
>
> Sent from my Samsung Galaxy smartphone.