

遠藤チームの研究成果のポスターが、GTC Japan 2014 で NVIDIA Award を受賞しました。

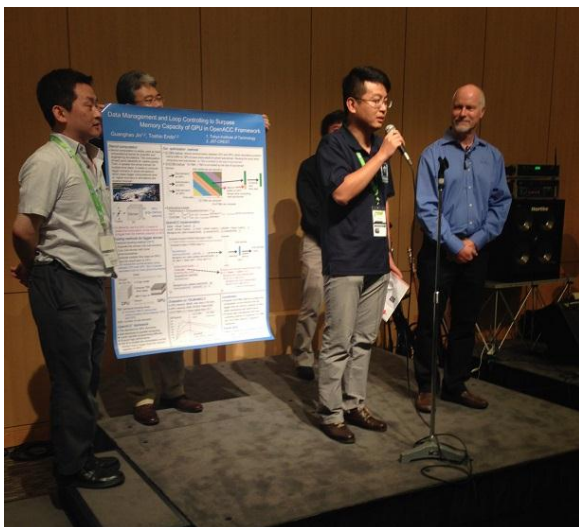
遠藤チームの CREST 研究成果のポスターが、東京で開催された GTC Japan 2014 において、NVIDIA Award を受賞しました。

受賞名: NVIDIA Award

受賞日: 2014 年 7 月 16 日

ポスター題目: Data Management and Loop Controlling to Surpass Memory Capacity of GPU in OpenACC Framework

受賞者: Guanghao Jin, Toshio Endo



Data Management and Loop Controlling to Surpass Memory Capacity of GPU in OpenACC Framework

Guanghao Jin^{1,2}, Toshio Endo^{1,2}

1. Tokyo Institute of Technology
2. JST-CREST

Stencil computation

- Stencil computation is widely used as base computational kernel for scientific and engineering simulations. The computation of each point depends on nearby points. Then, it updates the whole domain for multiple time steps. It needs to compute bigger domains in some simulations which mean bigger computational area or higher accuracy in simulations like weather forecast.

To efficiently use the GPU, it needs to enable the computation on the domain that is bigger than the memory capacity of GPU.

Existing methods for bigger domain

- Temporal blocking method (1D-T)
- Separate the domain into sub-domains.
- Copy sub-domain with more ghost boundaries.
- Compute multiple time steps on GPU.
- Copy the result back to CPU.

Can reduce the communication times between CPU and GPU. But, it causes redundant cost of more ghost boundaries.

Our optimization methods

- 1D-ETBM method:** reduce communication between CPU and GPU, solve redundant problem
- Using buffer on GPU to save some result of current sub-domain. Reusing the result when computing next sub-domain. TBS is limited by the size of sub-domain
- 1D-ETBM method:** 1D-TBM + TBS is not limited by the size of sub-domain

Performance models:

$$T_{1D-T} = T_{CPU} + C_{CPU} + C_{GPU} + C_{GPU} + C_{GPU} + C_{GPU} + C_{GPU} + C_{GPU}$$

$$T_{1D-ETBM} = T_{CPU} + C_{CPU} + C_{GPU} + C_{GPU}$$

OpenACC implementation

```

initial = float["calloc(...)"];
Grid0 = float["calloc(...)"]; Grid1 = float["calloc(...)"]; Buffer = float["calloc(...)"];
#pragma acc data create(Grid0, ...), create(Grid1, ...), create(Buffer, ...)
{
  for(Iter=0; Iter<ITERATION; Iter+=1)
  {
    if(TNSD==NSD) number of additional sub-domains
    for(step=0; step<=TNSD; step++)
    {
      Decide(loop);
      memcopy(&Grid0, ..., &Grid1, ...);
      #pragma acc data update device(Grid0, ...)
      for(Iter = Start; Iter != End; Iter++)
      {
        Buffer(Grid0, Buffer, ...);
        Compute(...);
        Buffer(Buffer, Grid0, ...);
        #pragma acc data update device(Buffer, ...)
        Decide(loop);
      }
      #pragma acc update host(Grid0, ...)
      memcopy(&Grid1, ..., &Grid0, ...);
    }
  }
}
    
```

Evaluation on TSUBAME2.5

- CPU memory: 56GB Intel Xeon 2.93 GHz
- GPU memory: 6GB, NVIDIA Tesla K20X
- 1D-ETBM is 2.7 times higher than 1D-T

Contribution

- Propose 1D-ETBM method to enable the computation on the domains that are bigger than the memory capacity of GPU while maintaining high performance.
- Propose data management and loop controlling method to implement 1D-TBM and 1D-ETBM method in OpenACC framework.

Future work

- Need to solve the consumption of space for buffer to implement the 1D-TBM and 1D-ETBM method.

To efficiently use the GPU, it needs to enable the computation on the domain that is bigger than the memory capacity of GPU.

Existing methods for bigger domain

- Temporal blocking method (1D-T)
- Separate the domain into sub-domains.
- Copy sub-domain with more ghost boundaries.
- Compute multiple time steps on GPU.
- Copy the result back to CPU.

Can reduce the communication times between CPU and GPU. But, it causes redundant cost of more ghost boundaries.

Our optimization methods

- 1D-ETBM method:** reduce communication between CPU and GPU, solve redundant problem
- Using buffer on GPU to save some result of current sub-domain. Reusing the result when computing next sub-domain. TBS is limited by the size of sub-domain
- 1D-ETBM method:** 1D-TBM + TBS is not limited by the size of sub-domain

Performance models:

$$T_{1D-T} = T_{CPU} + C_{CPU} + C_{GPU} + C_{GPU} + C_{GPU} + C_{GPU} + C_{GPU} + C_{GPU}$$

$$T_{1D-ETBM} = T_{CPU} + C_{CPU} + C_{GPU} + C_{GPU}$$

OpenACC implementation

```

initial = float["calloc(...)"];
Grid0 = float["calloc(...)"]; Grid1 = float["calloc(...)"]; Buffer = float["calloc(...)"];
#pragma acc data create(Grid0, ...), create(Grid1, ...), create(Buffer, ...)
{
  for(Iter=0; Iter<ITERATION; Iter+=1)
  {
    if(TNSD==NSD) number of additional sub-domains
    for(step=0; step<=TNSD; step++)
    {
      Decide(loop);
      memcopy(&Grid0, ..., &Grid1, ...);
      #pragma acc data update device(Grid0, ...)
      for(Iter = Start; Iter != End; Iter++)
      {
        Buffer(Grid0, Buffer, ...);
        Compute(...);
        Buffer(Buffer, Grid0, ...);
        #pragma acc data update device(Buffer, ...)
        Decide(loop);
      }
      #pragma acc update host(Grid0, ...)
      memcopy(&Grid1, ..., &Grid0, ...);
    }
  }
}
    
```

Evaluation on TSUBAME2.5

- CPU memory: 56GB Intel Xeon 2.93 GHz
- GPU memory: 6GB, NVIDIA Tesla K20X
- 1D-ETBM is 2.7 times higher than 1D-T

Contribution

- Propose 1D-ETBM method to enable the computation on the domains that are bigger than the memory capacity of GPU while maintaining high performance.
- Propose data management and loop controlling method to implement 1D-TBM and 1D-ETBM method in OpenACC framework.

Future work

- Need to solve the consumption of space for buffer to implement the 1D-TBM and 1D-ETBM method.