Utilization and Expansion of ppOpen-AT for OpenACC

Satoshi Ohshima, Takahiro Katagiri, Masaharu Matsumoto (The University of Tokyo)

contact: Satoshi Ohshima <ohshima@cc.u-tokyo.ac.jp>

INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

THE UNIVERSITY OF TOKYO

Background & Objective

- OpenACC makes GPU programming easy. Almost same source codes can be used on CPU, MIC, and GPU.
- However, to obtain good performance , users have to optimize programs by considering the architecture and characteristic of target hardware. Moreover, OpenACC programs have a lot of optimization parameters.
- Optimization of OpenACC is time-consuming and bother work.

- **ppOpen-AT** is a pragma-based Auto-Tuning (AT) launguage which provides various AT methods and adds AT functions to target applications.
- Because ppOpen-AT generates candidate codes and chooses the fastest one (semi-)automatically, **the cost of tuning will be reduced**. This is under development by ppOpen-HPC project. http://ppopenhpc.cc.u-tokyo.ac.jp/

In this study, we evaluate the effectiveness of ppOpen-AT for OpenACC program optimization and propose some expansion of ppOpen-AT for further OpenACC optimization.

Optimizing OpenACC programs by using ppOpen-AT

Major optimization strategies of OpenACC

1. Increasing a parallelism → make target loops long, collapsing the nested loops
2. Efficient memory access
3. Utilizing cache and register → modify the loop construction, concidering data structure (e.g. SoA/AoS)
4. Reducing data transfer → transfer really-required data, CPU vs GPU
5. Choosing processor

Example of Variable Selection feature

```
!OAT$ static variable (x) region start
!OAT$ name MyVal
!OAT$ varied (x) from 1 to 4
  ...
  blocksize = x
  ...
!OAT$ static variable region end
```

Performance of each value from a range of variables given with directives is measured and the best one value is chosen automatically.

Expansion: Variable Selection in Directive

```
!OAT$ static variableD (x) region start
!OAT$ name MyValD
!OAT$ variedD (x) from 32 to 512 step 32
!$acc kernels
!$acc loop gang vector (x)
  < target loop(s) >
!$acc end kernels
!OAT$ static variableD region end
```

To optimize OpenACC program, target value will be in directive. But, ppOpen-AT doesn' t touch directive. New feature modifies the variables in directives. This is useful for optimizing number of Gang/Worker/Vector, collapse level, and so on.

Expansion: Gang/Worker/Vector Optimization

To optimize Gang/Worker/Vector parameters are important for OpenACC. Variable Selection (in Directive) feature will be useful for this optimization, but it is difficult to indicate some special combinations of them simply. Therefore, we are developing special directives.

```
!OAT$ GWV-listed L (0,,256) (30,,128) (60,10,32)
!OAT$ hGWV-listed (L1(0,,),L2(,,64)) (L1(0,,),L2(,,128))
```

gang(60) worker(10) vector(32)

given lists of values are converted to the combinations of Gang/Worker/Vector

Kernel Selection feature

```
!OAT$ static select region start
!OAT$ name MySelect
!OAT$ select sub region start
  ... ! calculate on CPU
!OAT$ static sub region end
!OAT$ static sub region start
!$acc kernels copy(foo) ...
!$acc loop
  ... ! calculate on GPU
!$acc end kernels
!OAT$ static sub region end
!OAT$ static select region end
```

Kernel Selection feature is a general feature. Each sub regions are separated and chosen the fastest one automatically. In the case of OpenACC, choosing faster processor considering data transfer will be a useful use case.

Loop Transformation feature

```
!OAT$ static LoopFusionSplit region start
!OAT$ name MyTransform
!$acc kernels
!$acc loop
  < target loop(s) >
!$acc end kernels
!OAT$ static LoopFusionSplit region end
```

Loop Transformation feature generates collapsed, splitted, reordered, and the combinations of them versions of nested loops. And, the fastest one is chosen automatically.

Loop Unrolling feature

```
!OAT$ static unroll region start
!OAT$ name MyUnroll
!OAT$ varied (x) from 1 to 16
!$acc kernels
!$acc loop
  < target loop(s) >
!$acc end kernels
!OAT$ static unroll region end
```

Loop Unrolling feature specify the best unrolling level automatically. If memory access pattern will be effective, the performance may increase.

Current Issues

- Loop Transformation feature may generate wrong codes. For example, various OpenACC directives are inserted to inner loops and loop collapse confuses the relationships between directives and target loops. It will be common difficult issue for code modification tools not to generate wrong codes.
- Because OpenACC has various kind of directives and the combinations of them, the number of candidate code variations and time of performance measurement will increase. It is not an easy problem to know which is the best code variation quickly. ppOpen-AT has parameter search feature and it is expected that the feature make time of measurement short and correct.

Future Work

- evaluate more complex (real) programs (current target programs are sparse matrix calculation such as FEM, FDM, FVM, ...)
- compare the performance and trends with other processors (multi-core CPUs, many-core processor)
- propose and evaluate other useful features
- implement the translator completely

We believe that our next-gen' s ppOpen-AT will help various optimization works of OpenACC programs.