平成26年2月24日

「演算加速機構を持つ
将来のHPCIシステムに関する調査研究」
アプリ評価進捗報告

佐藤　三久

主管事業実施機関：　筑波大学
計算科学研究センター

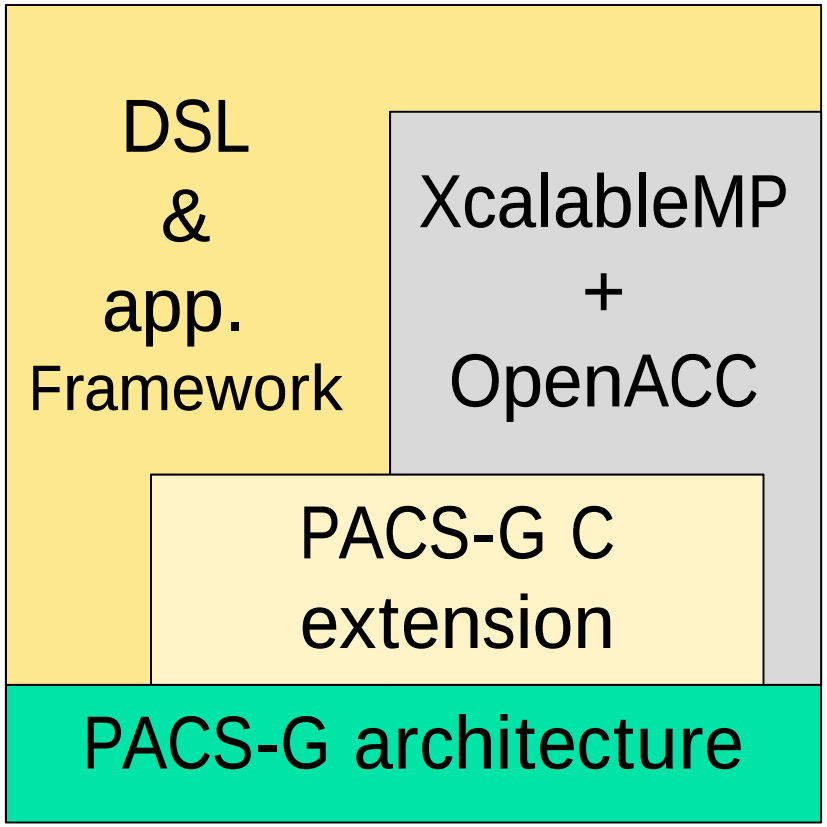
進捗・状況

- プログラミング環境（PACS-G　拡張C、 およびXcalableMP 拡張 PACS-G Fortran（実装中））と命令レベルシミュレータで、コードの準備
  - 性能評価に使うには、最適化をする必要あり
- サイクルレベルシミュレータ（アセンブラレベルのプログラム）

- 現在、1部サイクルレベルシミュレータ（地震波解析/stencil)と行列積等、のみで、アプリの評価はカーネルレベルの評価で行った。
- Strawmanから、2つのパラメータのマシンtypeA, typeBで評価。

Programming models for PACS-G

- PACS-G C extension for low-level programing
- XcalableMP (subset/extension) + OpenACC for directive based programming for stencil apps.
  - to make it easy to port existing codes
- Domain-Specific Language (DSL) and application framework
  - e.g. DSL for particle-based apps.
- (OpenCL?)



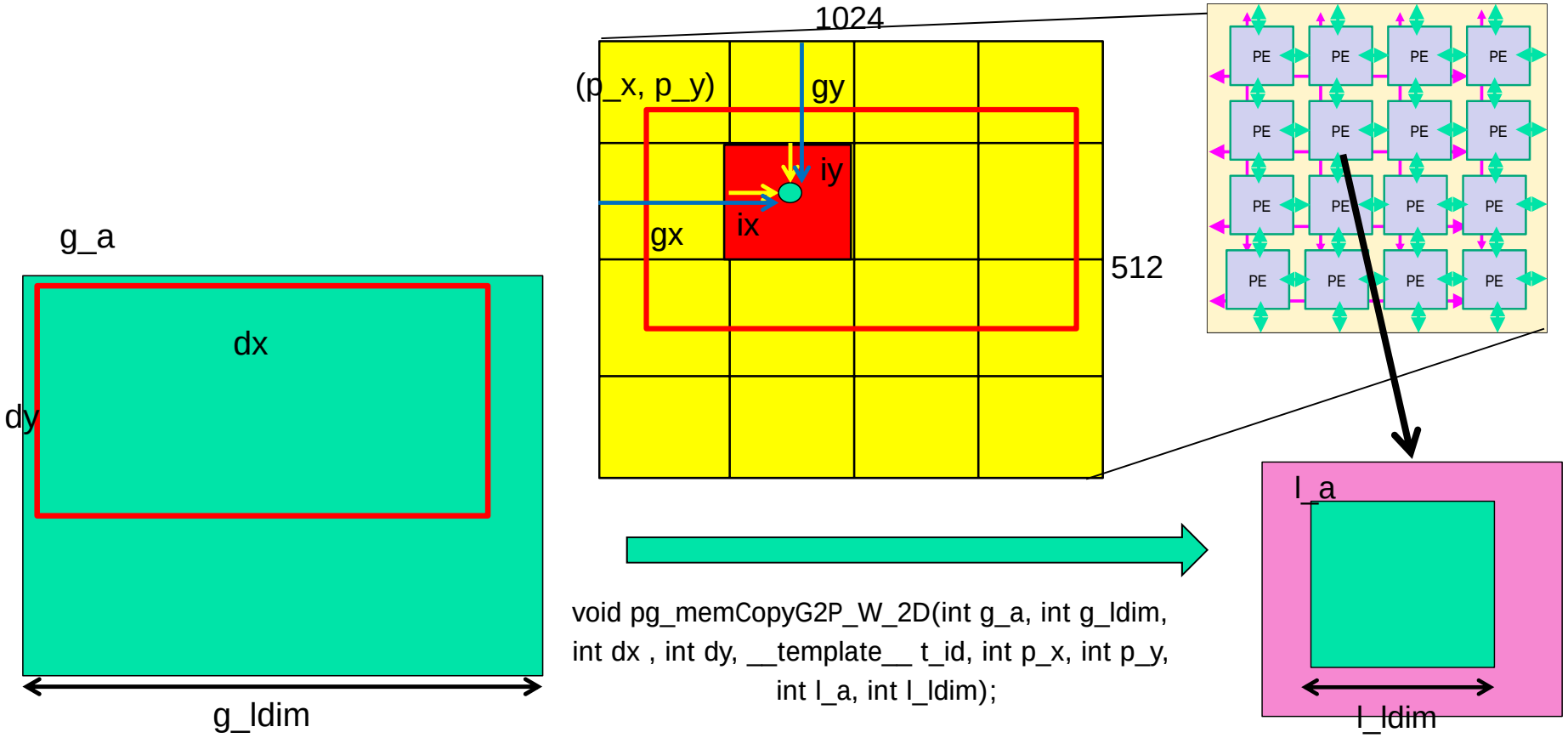
PACS-G C extension

- C extended for low-level programming of PACS-G SIMD architecture
- extended storage class to specify memory
  - global_memory: allocate data in global (module) memory
- __do_all__ statement
  - specify code fragments to be executed in PE
- function qualifiers:
  - __global__ : to allocate frame in PE
  - __all__ : functions executed in PE
- Intrinsic functions: compiled into instructions.
- template: index space over PEs
  - __for_all__ : parallel loop on template
- Host interface library

Template

- template: (virtual) index space over PEs
  - idea introduced in HPF and other data parallel lang (also in XMP)
  - __for_all__: parallel loop over PEs

    __for_all_ (template; lb1:ub1; lb2:ub2; ...) statement
    - register variables (ix, iy, ... gx, gy,...) gives local/global indices
  - also used to describe data transfer between PE and global memory



Code example (host code)

```
#include <stdio.h>
#include <PG_interface.h>
#define N 500
#define M 300

double A[M][N], B[M][N], C[M][N];

main()
{
    int i, r;  int x,y;
    void *g_a,*g_b,*g_c;

    PG_initialize("test13.exe");  // test4.c

    // allocate memory in GM
    PG_memAlloc(&g_a,sizeof(double)*N*M,PG_GMem);
    PG_memAlloc(&g_b,sizeof(double)*N*M,PG_GMem);
    PG_memAlloc(&g_c,sizeof(double)*N*M,PG_GMem);

    // copy out, input
    PG_memCopy(g_a,(char *)A,sizeof(double)*N*M,PG_memCopyHostToGMem);
    PG_memCopy(g_b,(char *)B,sizeof(double)*N*M,PG_memCopyHostToGMem);

    // call vectAdd on PACS-G
    r = PG_call("matAddGM",N,M,g_a,g_b,g_c);
    if(!r){   printf("call is failed¥n");  exit(1); }

    PG_memCopy((char *)C,g_c,sizeof(double)*N*M,
                             PG_memCopyGMemToHost);

    // free memory
    PG_memFree(g_a,PG_GMem);
    PG_memFree(g_b,PG_GMem);
    PG_memFree(g_c,PG_GMem);
}
```

note:
PACS-G proc can execute a main program without hosts

invoke a function on PACS-G proc

Code example (PACS-G)

```
#include <stdio.h>
#include <PACS_G.h>

void matAddGM(int n, int m, int *a, int *b, int *c)
{
    __template_t__ tmpl;
    int *l_a,*l_b,*l_c;
    int x_blk_siz, y_blk_siz, blk_siz;

    tmpl = pg_template2D(0,n-1,0,m-1);
    x_blk_siz = pg_templateBlockSize(tmpl,0);
    y_blk_siz = pg_templateBlockSize(tmpl,1);
    blk_siz = x_blk_siz*y_blk_siz;
    l_a = (int *)pg_pe_malloc(blk_siz*sizeof(double));
    l_b = (int *)pg_pe_malloc(blk_siz*sizeof(double));
    l_c = (int *)pg_pe_malloc(blk_siz*sizeof(double));

    pg_memCopyG2P_W_2D(a,n,n,m,tmpl,0,0,l_a,x_blk_siz);
    pg_memCopyG2P_W_2D(b,n,n,m,tmpl,0,0,l_b,x_blk_siz);

    matAdd(tmpl,n,m,x_blk_siz,l_a,l_b,l_c);

    pg_memCopyP2G_W_2D(c,n,n,m,tmpl,0,0,l_c,x_blk_siz);

    pg_pe_free(l_a);
    pg_pe_free(l_b);
    pg_pe_free(l_c);
}
```

```
/*
 * element-wise matrix add
 */

__global__ void matAdd(__template_t__ tmpl,
                       int n, int m, int w,
            double *l_a, double *l_b, double *l_c)
{
    int i,x,y;

    __for_all__(tmpl;0:n-1;0:m-1){
        x = __xi__; // local index
        y = __yi__;
        i = y*w+x;
        l_c[i] = l_a[i]+l_b[i];
    }
}
```

iterate on template (parallel loop)

copy from GM to PE using template

Fortran ホストインターフェース例

ftest4-host.f90

```
program ftest4
 double precision r,A,B,rr
 dimension A(100), B(100)
 integer :: a_addr, b_addr
 integer pg_symbol, pg_integer
 n = 100
 do i = 1,n
    A(i) = i+1
    B(i) = i+10
 enddo

 write(*,*) "test for pacs-g ..."
 call PG_initialize("g.out")  ! ftest4.f

 call pg_gm_malloc(a_addr,100*8)
 call pg_gm_malloc(b_addr,100*8)
 call pg_gm_malloc(r_addr,8)
 call pg_gm_memcopyout(a_addr,A,100*8)
 call pg_gm_memcopyout(b_addr,B,100*8)
 call pg_call(PG_symbol("inner_prod_"),4,r_addr,pg_integer(100),a_addr,b_addr)
 call pg_gm_memcopyin(rr,r_addr,8)
 write(*,*) "pg_call rr=",rr
end program ftest4
```

ftest4.f

```
subroutine inner_prod(r,n,A,B)
    integer n
    double precision r,A,B
    dimension A(n), B(n)
    r = 0.0
    do i = 1,n
       r = r + A(i)*B(i)
    enddo
    end
```

PACS-G Fortran XMP directive拡張

- 基本的には、OpenACCの拡張
  - データ配置を指定する。
  - templateを拡張して、PE/LMを仮想化
- template directive
  - PE上の仮想index空間を定義
  - distribution については、block分散のみを想定
- data directive
  - PEでのデータ環境を設定
  - データは当初は、GMまたはマスタ側にあることを前提とする。
  - copy/copyin/copyout/create/present/present_or_copy/present_of_copyin/present_or_copyout/prsesent_or_createにより、PE側にセットアップするデータを指定
- align directive / shadow directive
  - align により、templateに従い、分散配置。alignで指定されない場合には、それぞれのPEで重複。
  - alignで分散された配列に関しては、shadowで、袖領域を確保を指定
- parallel directive
  - 各PEで実行する部分を指定
  - __all__と同じ
- loop directive
  - ループを分割実行
  - on 節で、並列のindexスペースを指定。
  - reduction節で、reductionする変数を指定する。
- reflect directive
  - shadow領域のupdateを実行
- update directive
  - data環境中で、GM/マスタとPEのデータをやり取りする。

XMP/G Fortranによるプログラム

```
SUBROUTINE lap_main(xsize, ysize, u,uu)
integer: xsize, ysize
double dimension(0:xisze+1,0:ysize+1): u, uu
integer: x,y,k

!$xpg  template tmpl(0:XSIZE+1, 0:YSIZE+1)

!$xpg  data copy(u, uu)
!$xpg  align (i,j) with tmp(i,j) : u, uu
!$xpg  shadow  uu(1:1,1:1)
!$xpg  paralllel
      do k = 0, NITER
!$xpg array
         uu = u
!$xpgg reflect(uu)
!$xpgg loop on tmpl(i,j)
    do  x = 1, xsize
       do y = 1, ysize
         u(x,y) = (uu(x+1,y)+uu(x-1,y)+
                        uu(x,y+1)+uu(x,y-1))*0.25
       enddo
    enddo
  enddo
!$xpg end parallel
!$xpg end data
```

```
!$xpg parallel copyin(u,uu) copyout(sum)
!$xpg align (i,j) with tmp(i,j) : u, uu
      sum = 0.0
!$xpg loop on tmp(i,j) reduction(+:sum)
  do  x = 1, xsize
      do y = 1, ysize
           sum = (u(x,y) – uu(x,y))
     enddo
   enddo
!$xpg end parallel
...
if (sum .le. eps) ....

end subroutine lap_main
```

アプリ評価

- 次の2つのタイプを対象に評価
  - type A: 4096PE, LM 64KB/PE, 12.3TF/chip, 4096chip/group
  - type B: 2048PE, LM 128KB/PE, 8.2TF/chip, 4096chip/group
  - (Strawman): 4096PE, LM 128KB/chip, 16TF/chip
- 現在、評価中のアプリ
  - CCS-QCD＊　〇
  - Modylas　〇　<= ccp-MD＊
  - Seism3D (古村FDM？)　〇（△）　<= FDTD法＊
  - NICAM　△
  - Conquest　△
  - RS-DFT　△
  - N-Body＊(GreeM?)　〇
  - HMD＊　（宇宙磁気流体）　〇

△　精査中
＊昨年度

アプリ評価　報告

- CCS-QCD
  - 評価については、カーネルを人手で配置して解析
    - コードの準備状況
      - PACS-G拡張C言語での記述あり
      - PACS-G/XMP　Fortran準備中
- Modylas
  - 近距離計算のみ、長距離についてはホストで計算することを想定、
  - 現在、長距離力も加速機構で計算することを検討中

アプリ性能　報告

- RS-DFT
  - 6万4千原子。
  - 全計算時間の約60%以上を占めるDGEMM(2パターン)と約30%を占めるstencil(4次精度25点差分)をPACS-Gで実行することを想定.
  - 全てのデータはGM上に置き，必要に応じてBM/LMに転送しながら計算
  - コードの準備状況
    - 各部分のPACS-G向けコードは未完成であるが、方針は固まっている。
- N-body
  - 性能は、演算加速機構にオフロードされた重力計算のみを評価
    - ホストからの転送も含む
  - コードの準備状況
    - 拡張Cで、記述済み