Feasibility Study on Future HPC Infrastructure

-- Study on exascale heterogeneous systems with accelerators --

Mitsuhisa Sato

Professor, Center for Computational Sciences, University of Tsukuba /
Team Leader of Programming Environment Research Team, AICS, Riken

Outline

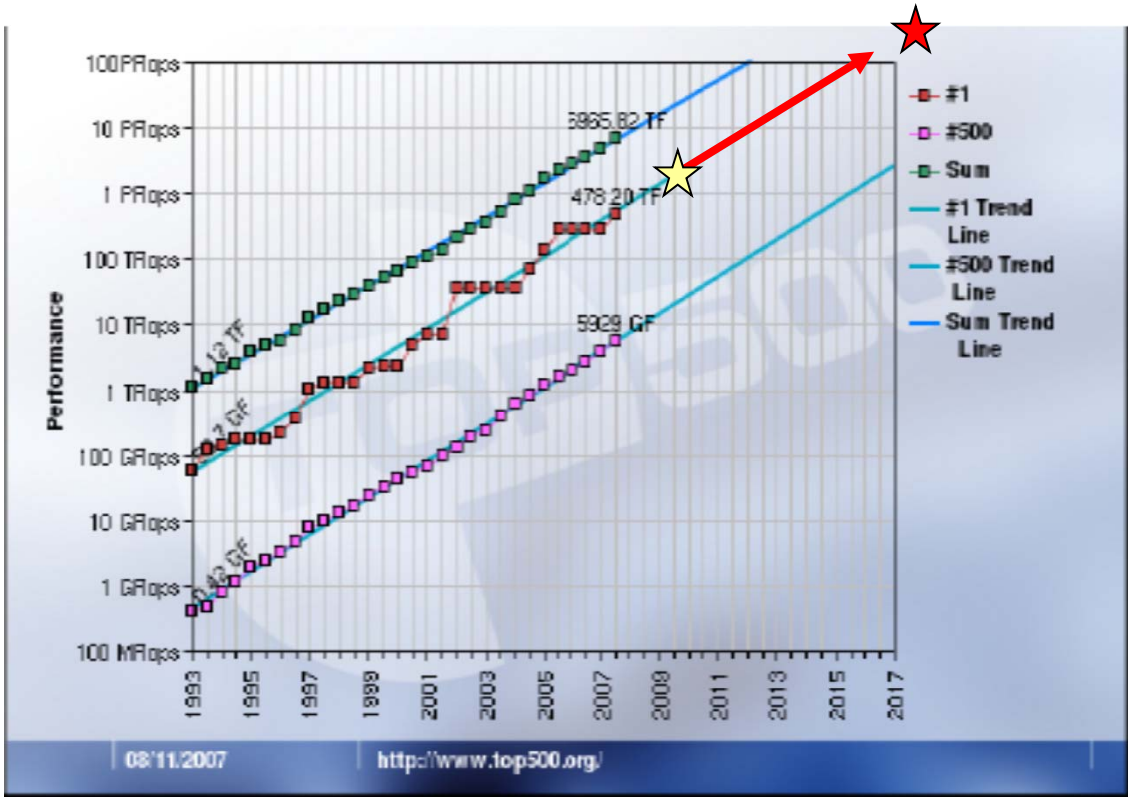
- Back ground: Issues for Exascale computing
  - Why Accelerated computing?
  - The SDHPC white paper and “Feasibility Study" project
- Our FS project: "Study on exascale heterogeneous systems with accelerators"
  - PACS-G: a straw man architecture
  - Performance estimation and Power estimation
  - Programming models for PACS-G
  - Current status and plan
- Summary and Concluding Remarks

Background: "Post-petascale computing“, toward exascale computing

- State of the art: Petascale computing infrastructure
  - US: Titan（27PF, 2012）,sequoia（>20PF,2012）
  - Japan: The K computer (>10PF, 2011), Tsubame 2.0
  - EU: PRACE machines (many of >5 PF, 2012-2013)
  - #cores 10^6
  - power >10 MW
- What's the next of "Petascale"?
  - Projection

  (and prediction)

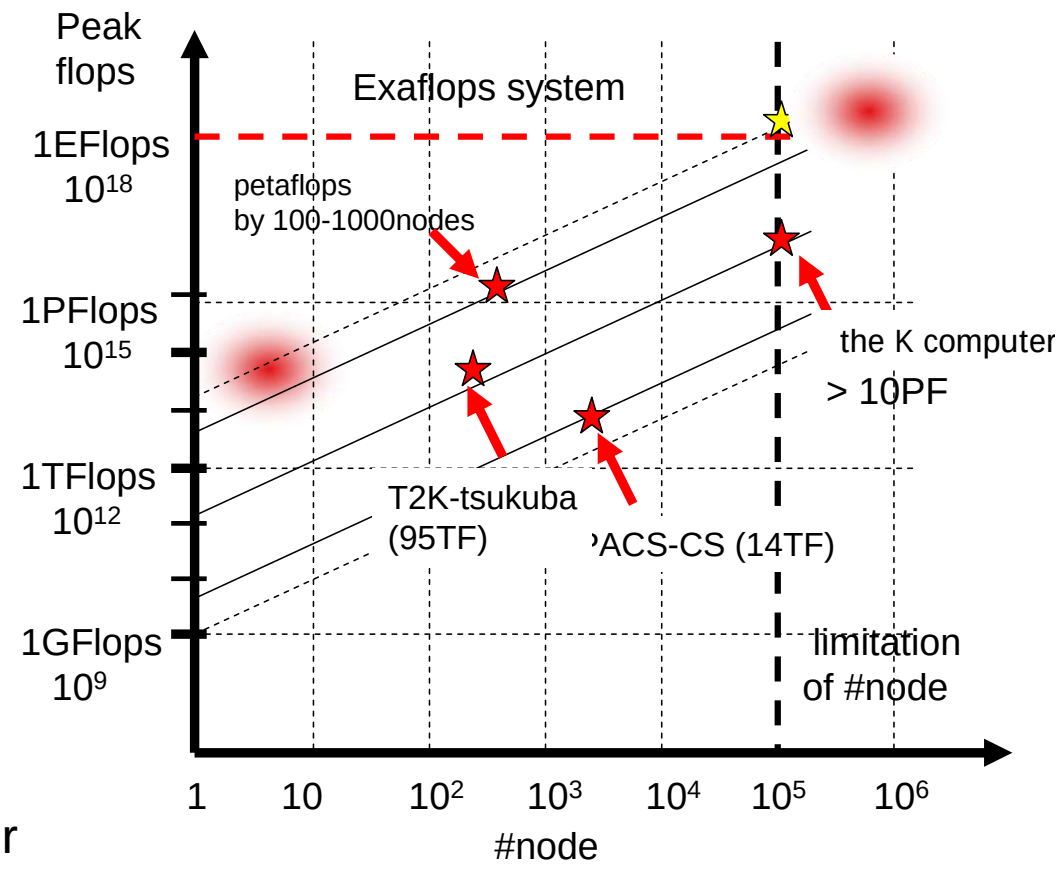
  by Top500



Issues for exascale computing

- Two important aspects of post-petascale computing
  - Power limitation
    - < 20-30 MW
  - Strong-scaling
    - < 10^6 nodes, for FT
    - > 10TFlops/node
    - accelerator, many-cores

- Solution: Accelerated Computing
  - by GPGPU
  - by Application-specific Accelerator
  - by ... future acceleration device ...



simple projection of #nodes and peak flops

The SDHPC white paper and Japanese “Feasibility Study" project

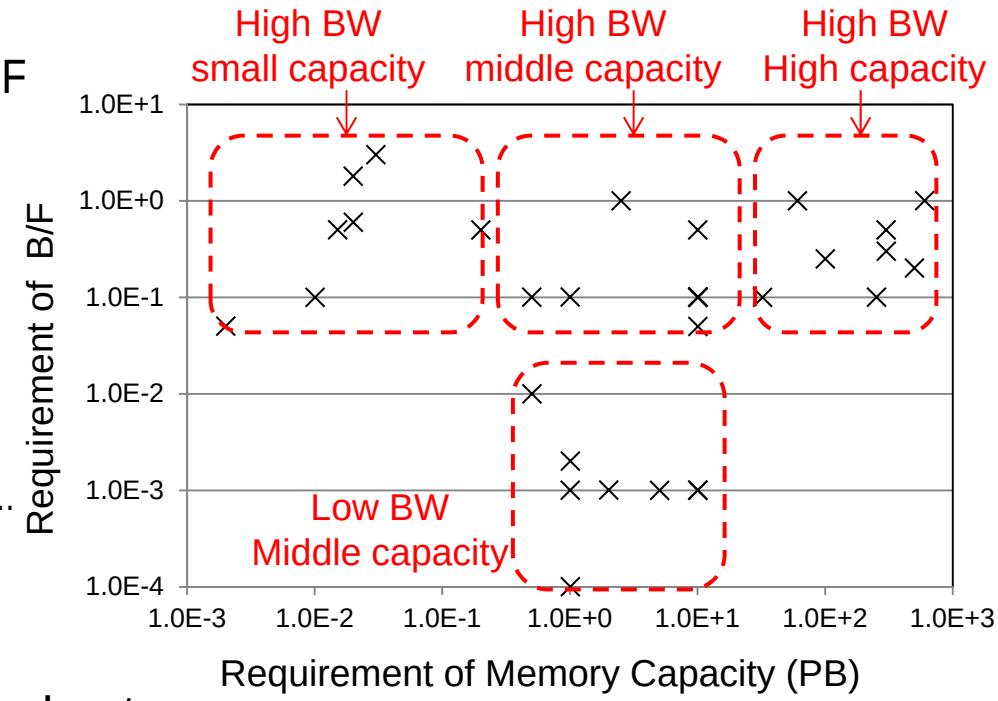
- WGs ware orgainzed for drafting the white paper for Strategic Direction/Development of HPC in JAPAN by young Japanese researchers with advisers (seniors)
- Contents
  - Science roadmap until 2020 and List of application for 2020’s
  - Four types of hardware architectures identified and performance projection in 2018 estimated from the present technology trend
  - Necessity of further research and development to realize the science roadmap

- For “Feasibility Study" project, 4 research teams were accepted
  - Application study team leaded by RIKEN AICS (Tomita)
  - System study team leaded by U Tokyo (Ishikawa)
    - Next-generation “General-Purpose” Supercomputer
  - System study team leaded by U Tsukuba (Sato)
    - Study on exascale heterogeneous systems with accelerators
  - System study team leaded by Tohoku U (Kobayashi)
- Projects were started from July 2012 (1.5 year) …

System requirement analysis for Target sciences

(From SDHPC white paper)

- System performance
  - FLOPS: 800 – 2500PFLOPS
  - Memory capacity: 10TB – 500PB
  - Memory bandwidth: 0.001 – 1.0 B/F
  - Example applications
    - Small capacity requirement
      - MD, Climate, Space physics, ...
    - Small BW requirement
      - Quantum chemistry, ...
    - High capacity/BW requirement
      - Incompressibility fluid dynamics, ...

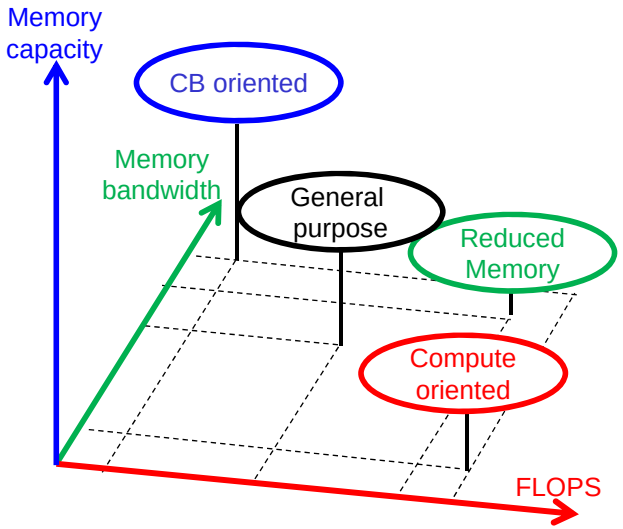


- Interconnection Network
  - Not enough analysis has been carried out
  - Some applications need >1us latency and large bisection BW
- Storage
  - There is not so big demand

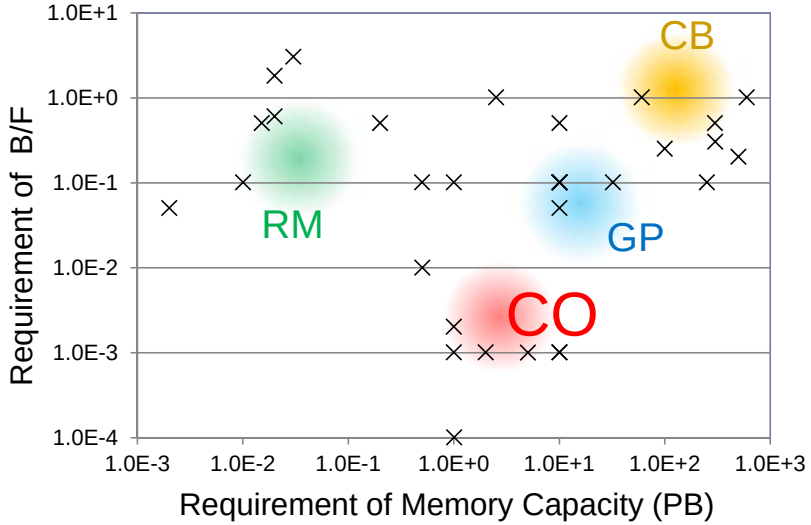
(From SDHPC white paper)

Alternatives of Exascale Architecture

- Four types of architectures are identified for exascale:
  - General Purpose (GP)
    - Ordinary CPU-based MPPs
    - e.g.) K-Computer, GPU, Blue Gene, x86-based PC-clusters
  - Capacity-Bandwidth oriented (CB)
    - With expensive memory-I/F rather than computing capability
    - e.g.) Vector machines
  - Reduced Memory (RM)
    - With embedded (main) memory
    - e.g.) SoC, MD-GRAPE4, Anton
  - Compute Oriented (CO)
    - Many processing units
    - e.g.) ClearSpeed, GRAPE-DR, GPU?

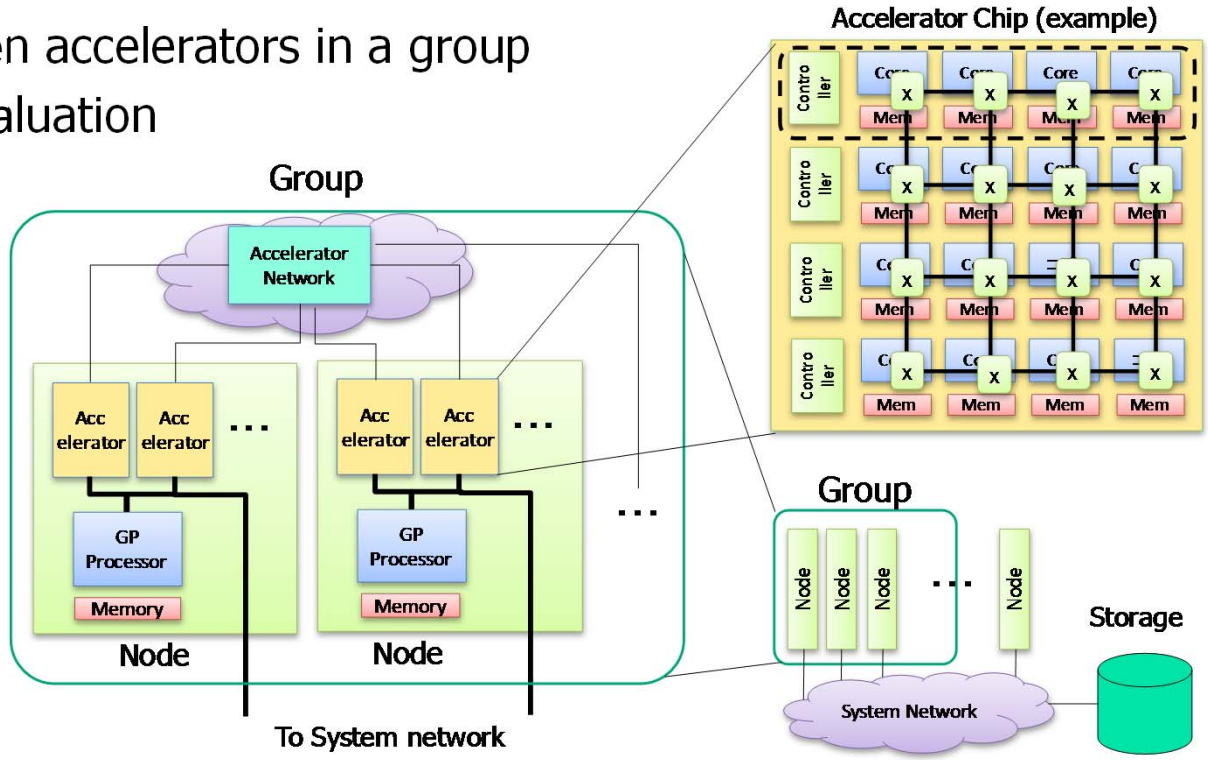


Mapping of Architectures



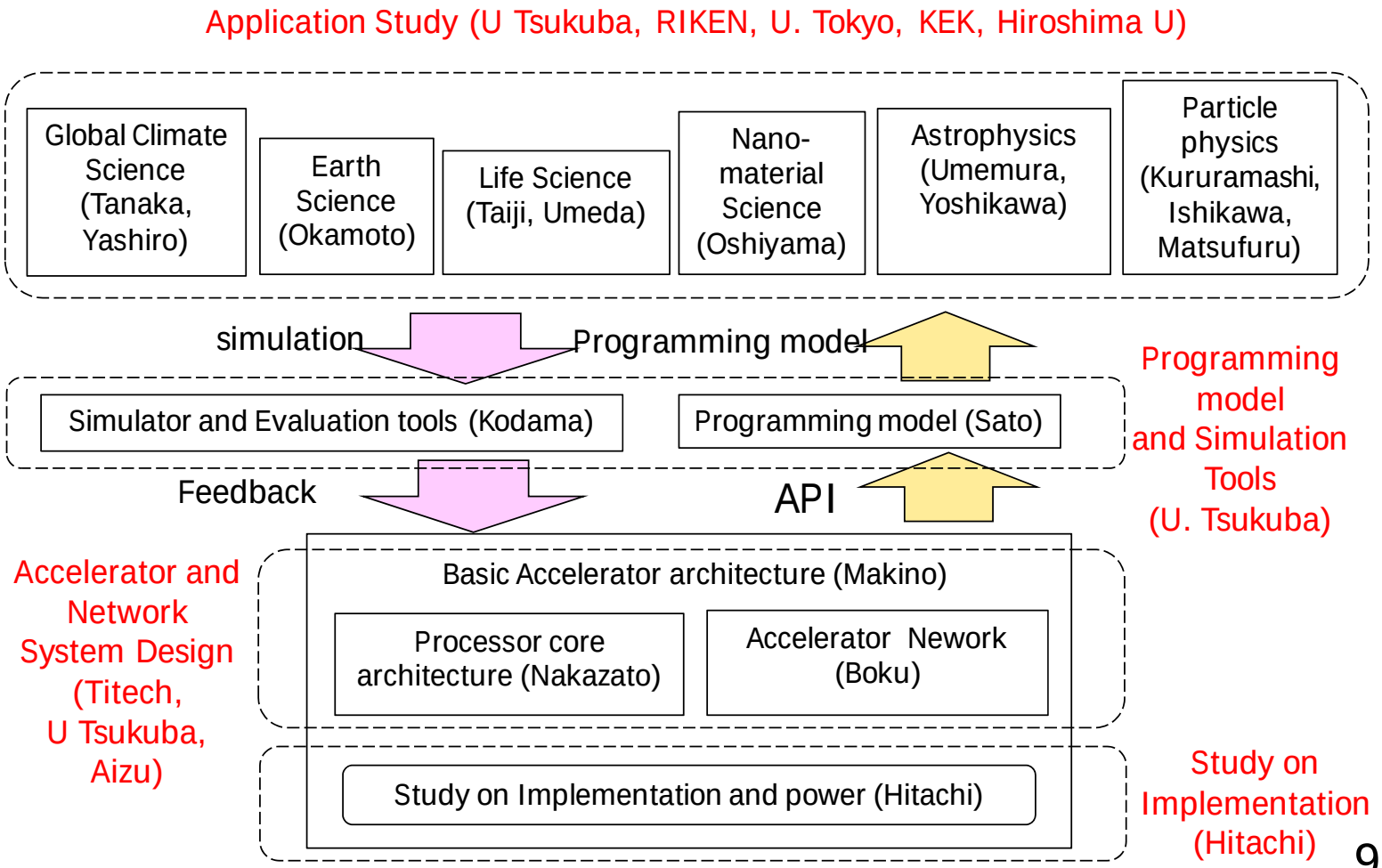
Study on exascale heterogeneous systems with accelerators (U Tsukuba proposal)

- Two keys for exascale computing
  - Power and strong-scaling
- We study “exascale” heterogeneous systems with accelerators of many-cores. We are interested in:
  - Architecture of accelerators, core and memory architecture
  - Special-purpose functions
  - Direct connection between accelerators in a group
  - Power estimation and evaluation
  - Programming model and computational science applications
  - Requirement for general-purpose system
  - etc ...



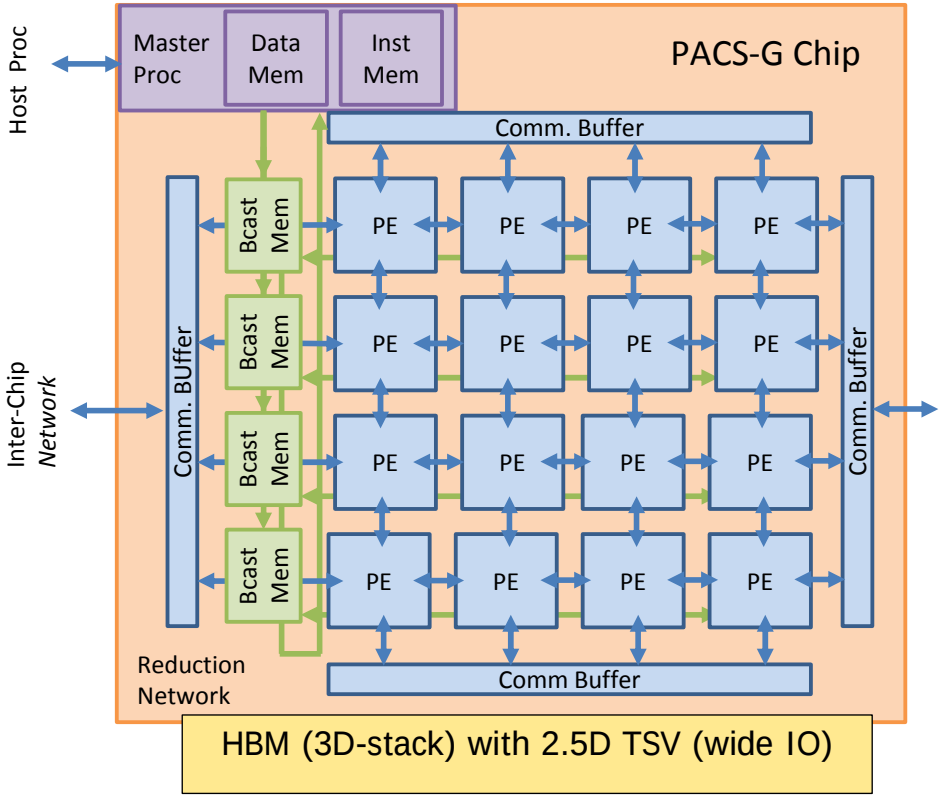
Project organization

- Joint project with Titech (Makino), Aizu U (Nakazato), RIKEN (Taiji), U Tokyo, KEK, Hiroshima U, and Hitachi as a super computer company
- Target apps: QCD in particle physics, tree N-body, HMD in Astrophysics, MD in life sci., FDM of earthquake, FMO in chemistry, NICAM in climate sci.



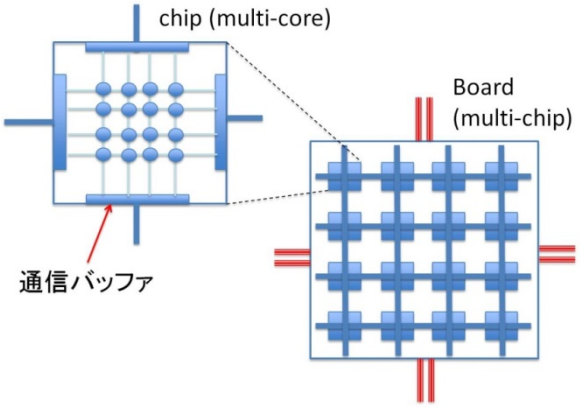
PACS-G: a straw man architecture

- SIMD architecture, for compute oriented apps (N-body, MD), and stencil apps.
- 4096 cores (64x64), 2FMA@1GHz, 4GFlops x 4096 = 16TFlops/chip
- 2D mesh (+ broadcast/reduction) on-chip network for stencil apps.
- We expect 10nm technology available in the range of year 2018-2020, Chip-dai size: 20mm x 20mm
- Mainly working on on-chip memory (size 512 MB/chip, 128KB/core), and,
- with module (global) memory by HBM (3D-stack/wide IO DRAM memory via 2.5D TSV), bandwidth 1TB/s, size 16-32GB/chip (block access only, no random access)
- No external memory (DIM/DDR)
- 250 W/chip expected
- 64K chips for 1 EFLOPS (at peak)

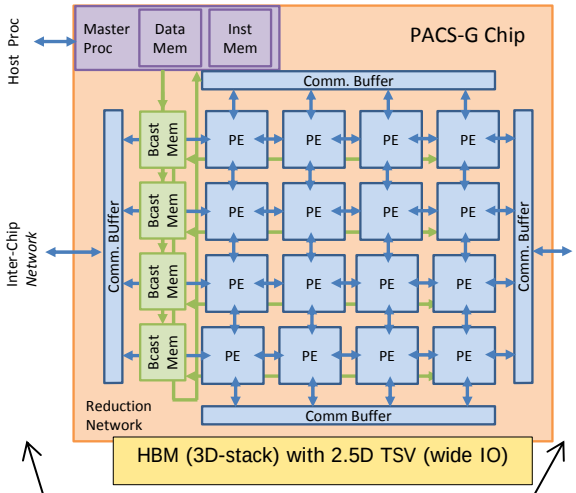


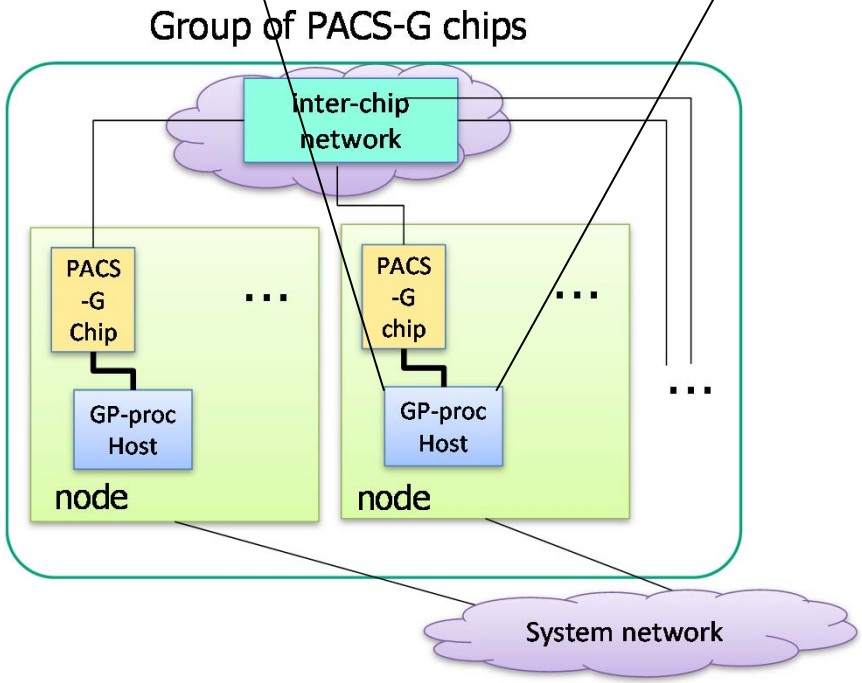
PACS-G: a straw man architecture

- A group of 1024～2048 chips are connected via accelerator network (inter-chip network)
- 25 – 50Gpbs/link for inter-chip: If we extend 2-D mesh network to the (2D-mesh) external net work in a group, we need 200～400GB/s (= 32 ch. x 25～50Gbps x 2(bi-direction))⇒too much!
- For 50Gpbs data transfer, we may need direct o ptical interconnect from chip.
- I/O Interface to Host: PCI Express Gen 4 x16 (not enough!!!)



interconnect between chips (2D mesh)





Performance estimation of some applications

- We did performance estimation (upper bound) of some applications with counts of calculation(FLOPS) and memory access (bandwidth), pattern of communications on a group of PACS-G proc (upto 2048 procs, 32PF)
  - When all data fits on on-chip memory, ratio B/F is 4 B/F, total mem size 1TB/group
  - When data fits into module memory, ratio B/F is 0.05B/F, total mem size 32TB/group

application	Expected problem size	Efficiency / performance	comments
Lattice QCD (particle physics)	Physical volume(12fm)^4 Hydron Manybody System 128^4 lattice	12%～53% 7.9～34.7PF 2048proc(single precision peak 65.5 PF)	• 評価対象アルゴリズム：領域分割前処理単精度クォークソルバー（ウィルソンクォーク型、BiCGStab法） • Use only on-chip memory • Communication latency is significant. eps. for global reduction • On K computer: efficiency 26%, 32768 nodes, 1.1 PF
Hydro-Magnetic Dynamics code (astrophysics)	Number of cells 1984^3	1.89 PF, 22.5% 512 proc(8PF)	• HLL近似リーマン解法、磁場をflux-CT法よる有限体積法。時間積分を2次精度のTVD Runge-Kutta法 • Use global(module) memory • 210-220ms/step, 4.5s/step in case of Intel Core i7 4096 core
Gravity N-body (astrophysics)	814G interaction/sec/chip （single preceision, no-collision）		• Offload only gravity calculation, assuming trajectory computation is performed by host. • Use only on-chip meomry • 66.7 times faster that Intel Xeon E5-2670
Molecular Dynamics（MD） Kernel (Life science)	1cell/core、1 cell (5Å)^3, cut-off radius 12Å 2580 atoms/core	3.67PF、 Max 15Matoms /256 proc, 784.4us/step	• Note only short-distance force is considered. 遠距離相互作用計算、結合力計算は未評価 • セルインデックス法（空間座標分割）とハーフシェルスキームを仮定 • On K computer, 500M atoms using whole system、4.6PF, 114ms/step
Seismic wave computation(earth science)	Lattice size 2048x2048x512	3.5 PF /1024 procs	• 3-D Finite-Difference Time-Domain methodtime（FDTD）、空間差分4次精度、時間差分2次精度、弾性体、速度と応力を変数とするスキーム • Use Only on-chip. global memory will be used

Power-consumption estimation and Revision from strawman

- Inter-chip network:
  - 22.4 GB/s (8 lanes/ports) x 6 ports may be reasonable (enough?) for a wide range of applications.
- Clock frequency of processor:
  - To optimize performance and power, 800MHz-700MHz will be adequate rather than > 1GHz. We need more investigation for trade-off of of cores (PE) and clock freq.

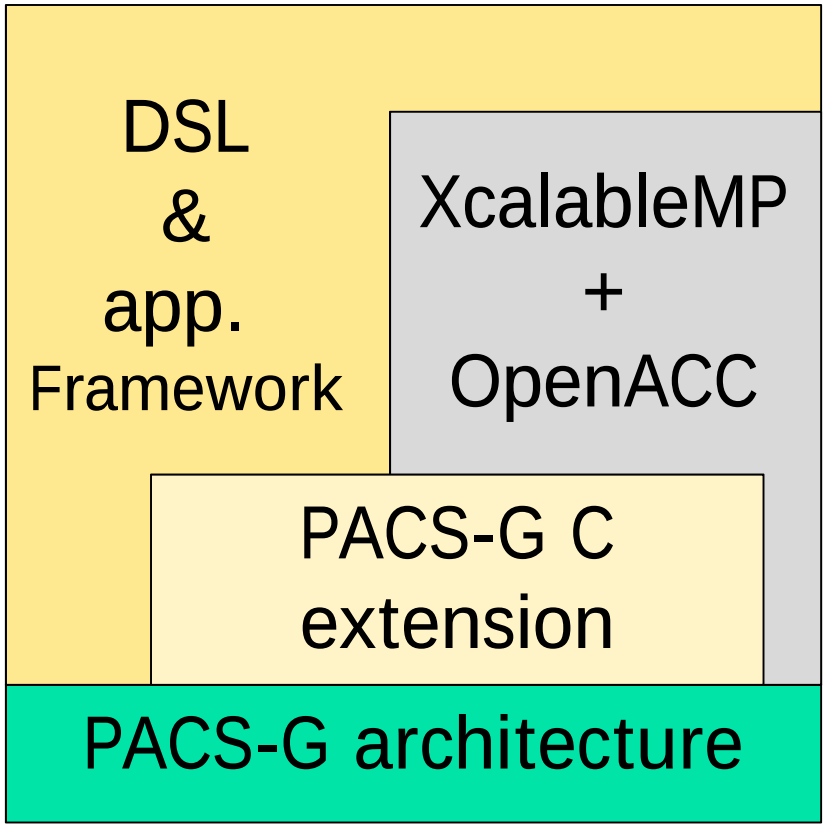
Elements	Power (W)
Processor Chip (#PE = 5120)	96.3
Power per PE	0.0188
Module Memory	
2.5D TSV, HBM 8GB x 4=32GB	36.0
HMC 16GB x 4	60.0
Inter-chip network	14.4
misc (DC/DC, AC/DC overhead, etc)	+ 25%
Total power consumption per proc	

This estimation should be revised for 10nm technologies

- Power consumption and memory bandwidth of module memory:
  - If we can use HBM with 2.5D TSV, 1TB/s , 70GF/W, (otherwise HMC)
  - 750MHz, 1FMA/.core, 10240cores(1.53PF) , 2.5D HBM 32GB ⇒183W
  - 750MHz, 1FMA/.core, 10240cores(1.53PF) , HMC 64GB ⇒ 213W
- We need more study for space (chip dai-size) …(with 10nm FinFET technology)

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?)



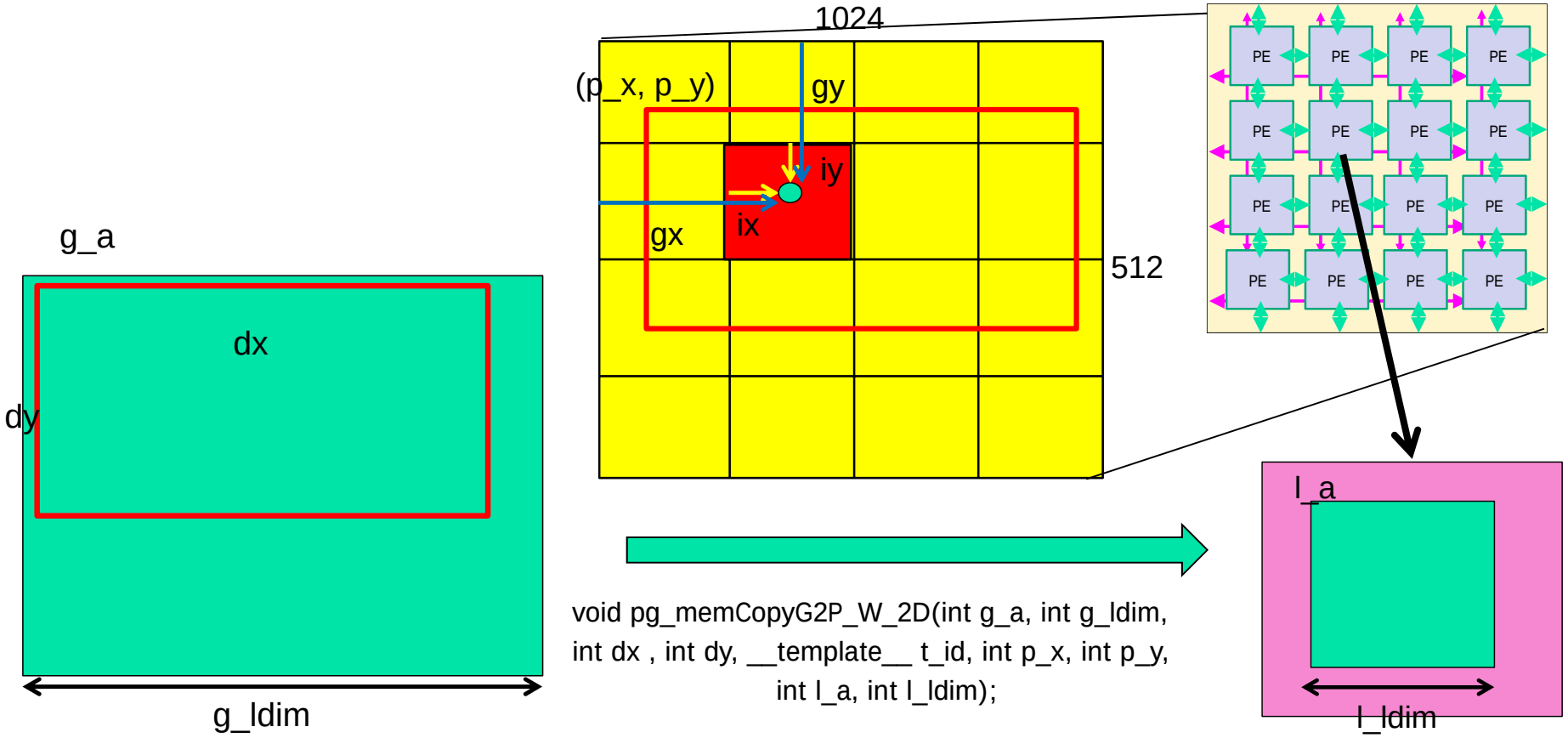
PACS-G C extension

- 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);
}
```

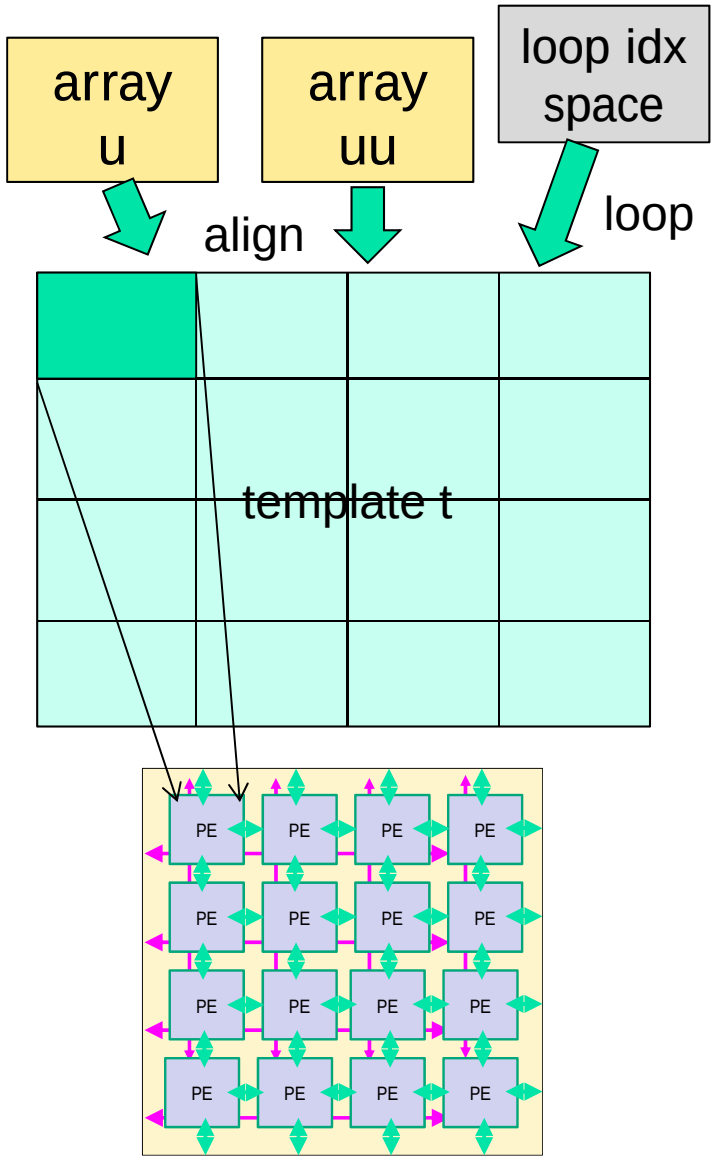
```
/*
 * element-wise matrix add
 */

__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

Programming model: XcalableMP + OpenACC

- Use OpenACC to specify offloaded fragment of code and data movement
- To align data and computation to core, we use the concept "template" of XcalableMP (virtual index space). We can generate code for each core.
- It will be useful to port existing stencil code to PACS-G architecture



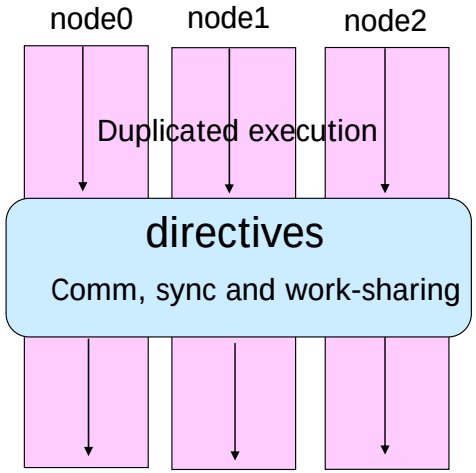

```
#pragma xmp template t(0:XSIZE+2, 0:YSIZE+2)

double u[XSIZE+2][YSIZE+2],uu[XSIZE+2][YSIZE+2];
#pragma xmp align u[i][j] with t(i,j)
#pragma xmp align uu[i][j] with t(i,j)
#pragma xmp shadow uu[1:1][1:1]

#pramga xmp reflect uu
#pragma xmp loop on t(x,y)
  for(x = 1; x <= XSIZE; x++)
   for(y = 1; y <= YSIZE; y++)
      u[x][y] = (uu[x-1][y] + uu[x+1][y]
              + uu[x][y-1] + uu[x][y+1])/4.0;
sum = 0.0;
#pragma xmp loop on t(x,y) reduction(+:sum)
  for(x = 1; x <= XSIZE; x++)
    for(y = 1; y <= YSIZE; y++)
        sum += (uu[x][y]-u[x][y]);
```

XcalableMP

http://www.xcalablemp.org

XcalableMP : directive-based language eXtension
for Scalable and performance-aware Parallel Programming

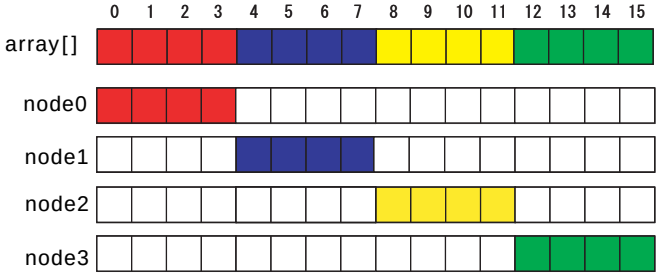
- A PGAS language. Directive-based language extensions for Fortran95 and C99 for the XMP PGAS model
  - To reduce the cost of code-rewriting and education
- Global view programming with global-view distributed data structures for data parallelism
  - A set of threads are started as a logical task. Work mapping constructs are used to map works and iteration with affinity to data explicitly.
  - Rich communication and sync directives such as “gmove” and “shadow”.
  - Many concepts are inherited from HPF
- Co-array feature of CAF is adopted as a part of the language spec for local view programming (also defined in C).




```
int array[N];
#pragma xmp nodes p(4)
#pragma xmp template t(N)
#pragma xmp distribute t(block) on p
#pragma xmp align array[i][ with t(i)

#pragma xmp loop on t(i)  reduction(+:res)
  for(i = 0; i < 10; i++)
     array[i] = func(i,);
     res += …;
   } }
```

Global view programming for data-parallel stencil apps. in XMP

- The following directives specify a data distribution among nodes
  - #pragma xmp nodes p(*)
  - #pragma xmp template T(0:15)
  - #pragma xmp distribute T(block) on p
  - #pragma xmp align array[i] with T(i)

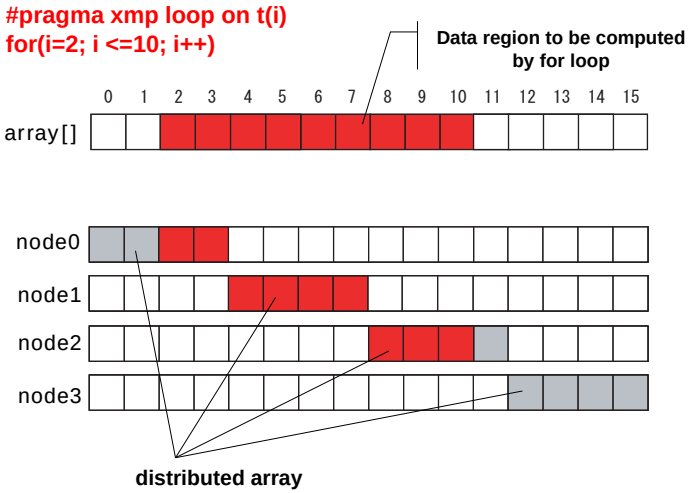


data distribution and allocation

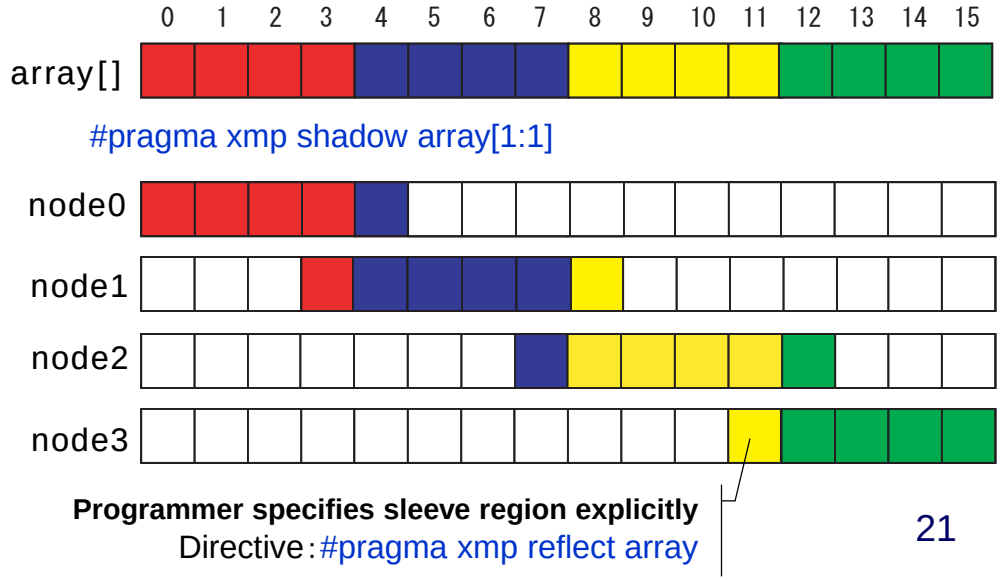
parallel loop aligned to data

neighbor communication

- Execute for loop to compute on array



- Exchange data only on "shadow" (sleeve) region
  - If neighbor data is required to communicate, then only sleeve area can be considered.
  - example : b[i] = array[i-1] + array[i+1]



Current status

- Performance estimation by co-design process
  - 2012 (done): QCD, N-body, MD, HMD
  - 2013: earth quake sim, NICAM (climate), FMO (chemistry) ⇒ RSDFT, Conquest
- Development of simulators (clock-level/instruction level) for more precious and quantitative performance evaluation

- Programming models and compiler development
  - PACS-G C extension
  - XMP for PACS-G (and OpenACC)
- (Re-)Design and investigation of network topology
  - For both on-chip network and inter-chip network
  - 2D mesh is sufficient? or, other alternative?
- Precise and more detail estimation of power consumptions
- Application development for PACS-G for quantitative performance evaluation, using our programming models. ...
- .... Further study on the architecture for beyond stencil and particle apps.

Summary and Concluding Remarks

- Issues for exascale computing
  - Power and Strong-scaling
  - Solution: Accelerated Computing

- PACS-G: the "extreme SIMD" architecture
  - Co-design for compute oriented apps (N-body, MD), and stencil apps.
  - Aiming to high performance(> 10TF/chip) , but also to good performance/power.
  - On-chip memory for each PE using 10nm silicon technology at 2018-2020.
  - dedicated inter-chip network and HBM with 2.5 TSV
  - Programming models and applications are under development for performance evaluation.

- *I believe acceleration with the "extreme SIMDs" and on-chip memory is a first (and good) candidate for exascale computing in some scientific fields.*

- Our FS project will be over at the end of 2013FY.

- We are proposing our architecture for Japanese "exascale" supercomputer, which development will be conducted by Riken AICS from 2014FY.
  - The national "exascale" supercomputer project is aiming a "exascale" system in 2019-2020.