A report on Feasibility Study on Future HPC Infrastructure

"Study on exascale heterogeneous systems with accelerators"

PACS-G architecture and system overview

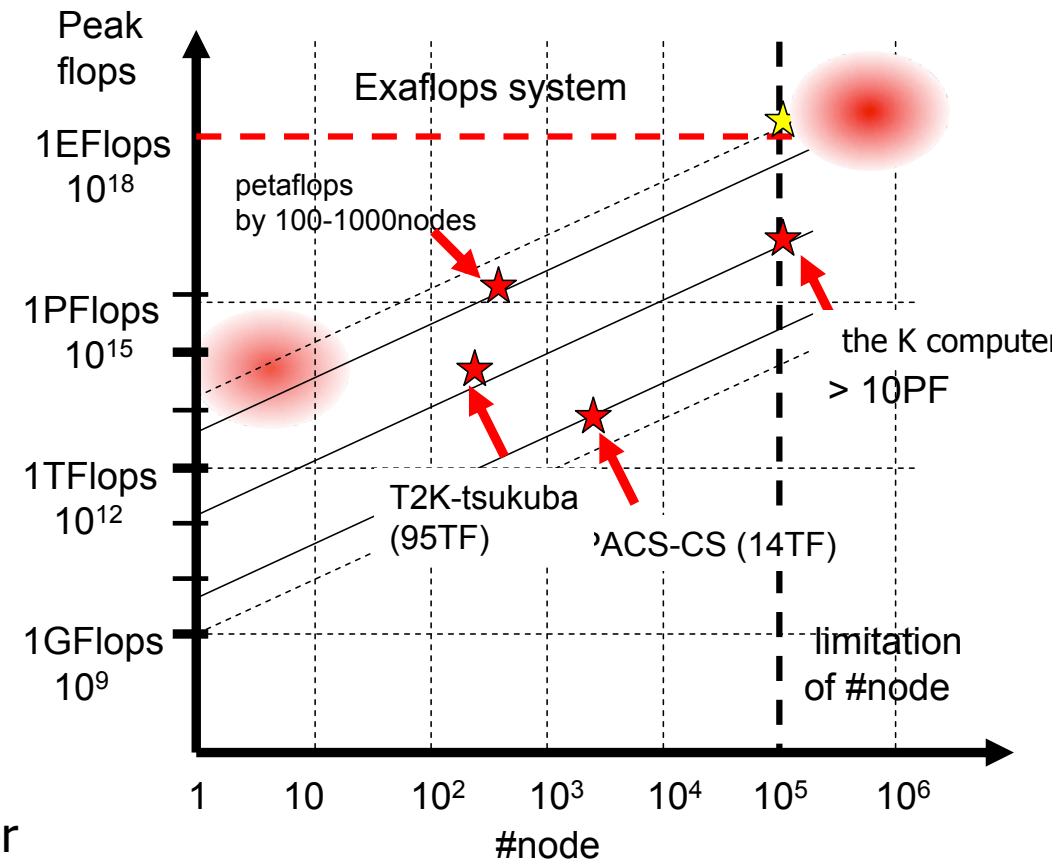
Mitsuhisa Sato

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

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

京コンピュータ “The K computer"

- cabinet 864
- chips 82,944
- core 663,552
- performance Linpack n10.51PF (power 12.66MW)

The SDHPC white paper and Japanese "Feasibility Study" project

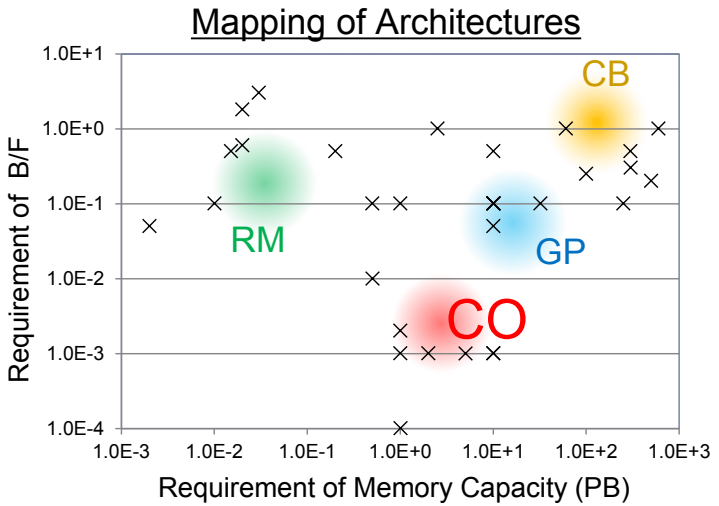
- WGs ware organized for drafting the white paper for Strategic Direction/ Development of HPC in JAPAN by young Japanese researchers with advisers (seniors) in 2010-2011
- Contents
  - Science roadmap until 2020 and List of application for 2020's
  - Four types of hardware architectures identified and 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)
    - "Memory Bandwidth-oriented" architecture.
- Projects were started from July 2012 (1.5 year) …

Objectives of our HPCI-FS project

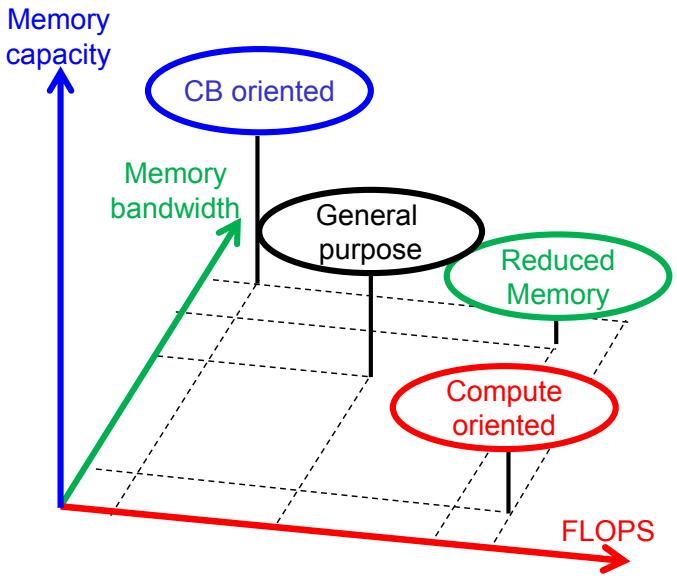
- Demand of computing power for advanced computational sciences is increasing
  - Computational Science is a critical and cutting-edge methodology in all of science and engineering disciplines including nano-tech, life science, climate and weather prediction, disaster prevention and mitigation
  - Space and power consumption limits the whole system performance by increasing the number of nodes
- On other hand, many applications such as MD simulation in life science needs speedup for fixed size of problems in real-time (Strong scaling)
  - example: ANTON, MDGRAPE-4

（From SDHPC white paper）



To enable new computational sciences by Significant Improvement of power-performance efficiencyとSpeedup with strong scaling, we carry out feasibility study on large scale parallel system with accelerators

- Among architectures identified in SDHPC white paper, our feasibility study covers Reduce Memory (RM) and Compute Oriented (CO)



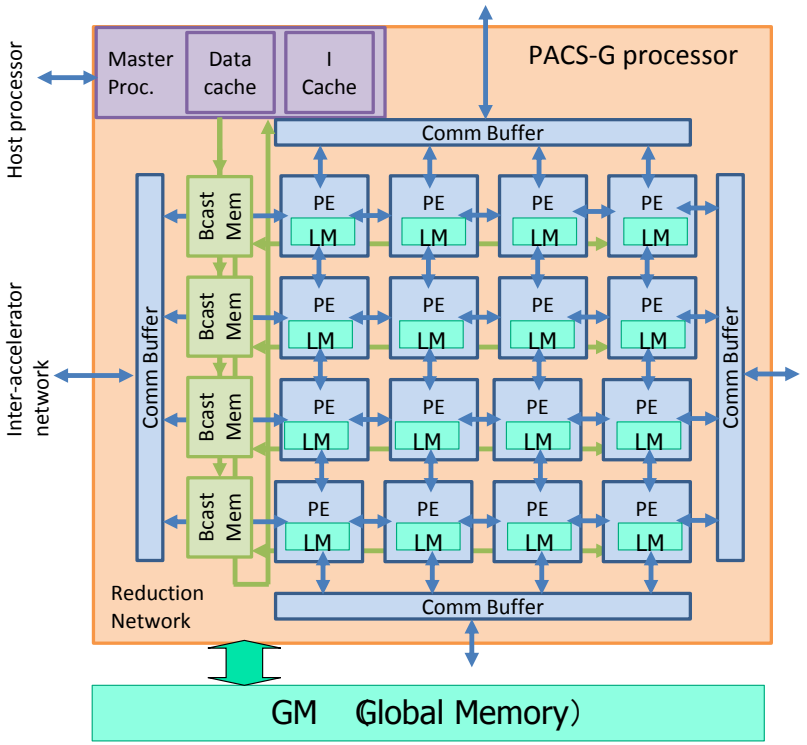
- Speedup with Strong Scaling will make MD applications significantly faster in real-time.
- Significant Improvement of power-performance efficiency will make large-scale execution of reduced-memory type of applications such as lattice QCD more efficiently with smaller amount of energy.

PACS-G Architecture: Overview （1）

Summary:

- Our target types of applications are both Compute-oriented type (N-body, MD, dgemm) and Reduced-memory type stencil computations.
- The accelerator co-processor (PACS-G) consists of large number of PE(core), which is controlled in SIMD manner ("extreme" SIMD architecture). This architecture enables significant improvement of FLOPS by large number of PEs, simplification of controlling PEs, and significant reduction of energy for computation.
- PE consists of arithmetic units, registers and local memory (LM). The instruction operates on LM. PEs are connected by on-chip network.
- The co-processor are connected via dedicated inter-chip network, which enables low latency communication to realize efficient parallel execution with strong scaling.
- We expect 10nm (FinFET) LSI technology available in year 2018-2020, chip-dai size: 20mm x 20m

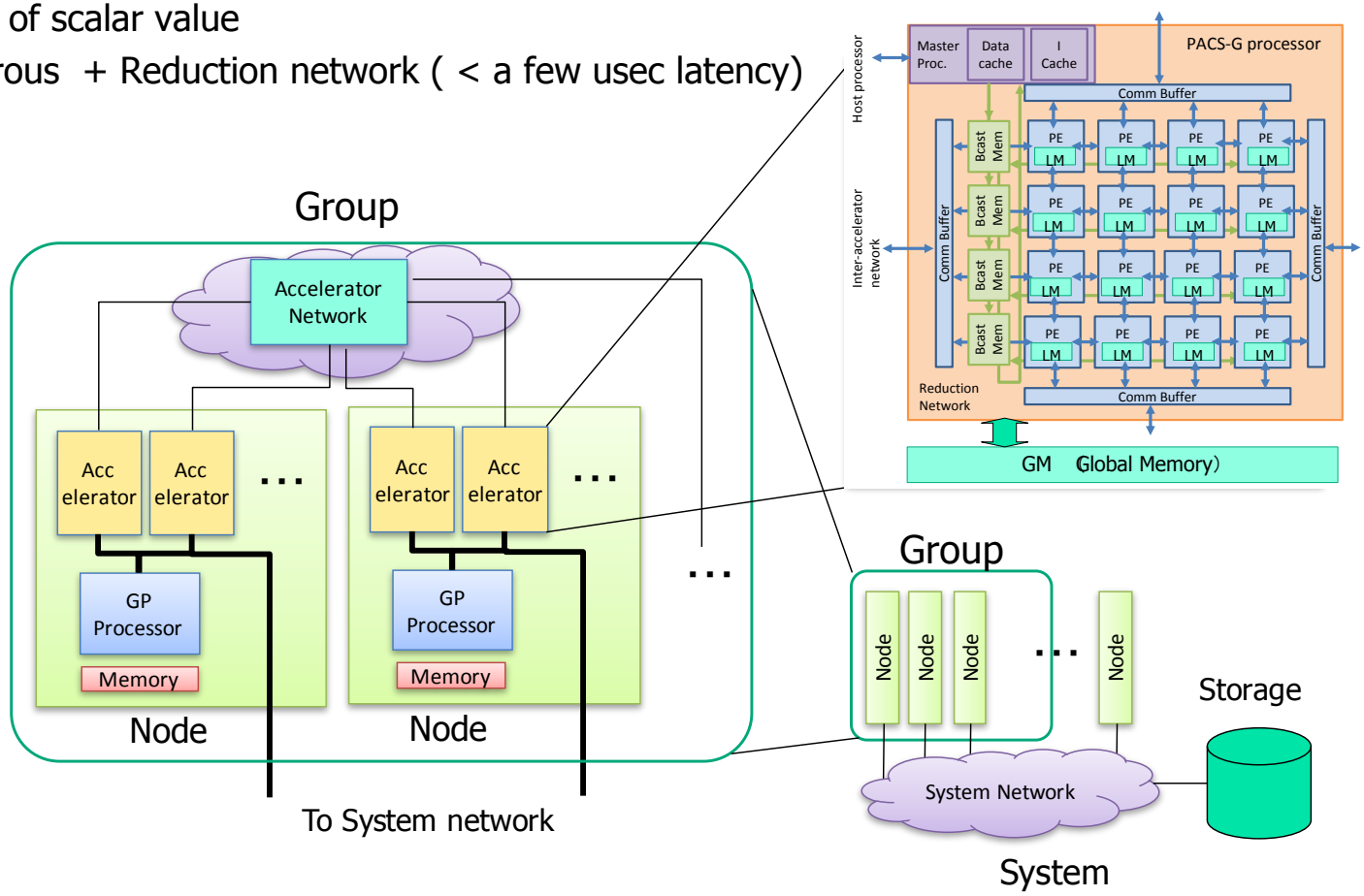
- Master processor may be conventional letency-core. PE is controlled as extended instructions in the master processor.
- We estimate the number of PEs 2048～4096 per chip.
- We assume external "global memory" (GM) to the chip.
  - High-Bandwidth Memory (HBM, TSV, 2.5 Stacking) or HMC (Hybrid <emory Cube)
  - DDR/DIM cannot be used.
  - Only block data transfer between PE to GM. No random access may be supported.
- On-chip network may support neighbor communication of PEs upto 4-demension mesh topology (Note: Figure only shows 2D-mesh)
- Support On-chip network and dedicated memory for reduction and broadcast to PEs.
- Target performance-power efficiency > 50GF/W (10nm)



PACS-G Architecture: Overview （2）

- Each node consists of general purpose processor with a few accelerator co-processor (PACS-G)
- Each node are connected via system network.
- Co-processor are connected via PCI-e
  - I/O Interface to Host: PCI Express Gen 4 x16 (32GB/s)　(not enough!!!)

- A group of 2048～4096 chips are connected via accelerator network (inter-chip network)
- Inter-chip network may support neighbor communication upto 4D topology and reduction operation of scalar value
  - Current plan:　4D-trous　+ Reduction network (< a few usec latency)



PACS-G architecture: Pros and Cons for exascale

- Low power (target > 40-50GF/W) & Strong scaling
  - "extreme" SIMD, less power, and easy to synchronize with out overhead
  - Local memory for each PE.
  - block transfer from GM to LM.
  - ... But, less flexibility and difficult to program ...
  - direct network between processors for shorter latency.

- Comparing to GPU
  - power efficiency < 30GF/w ...
  - no direct network
  - ..., but many software is getting matured.

- Comparing to manycore
  - power efficiency < 20GF/W
  - Huge overhead to synchronize million of threads.
  - ..., but many software already exists!!!

Issues of software for PACS-G

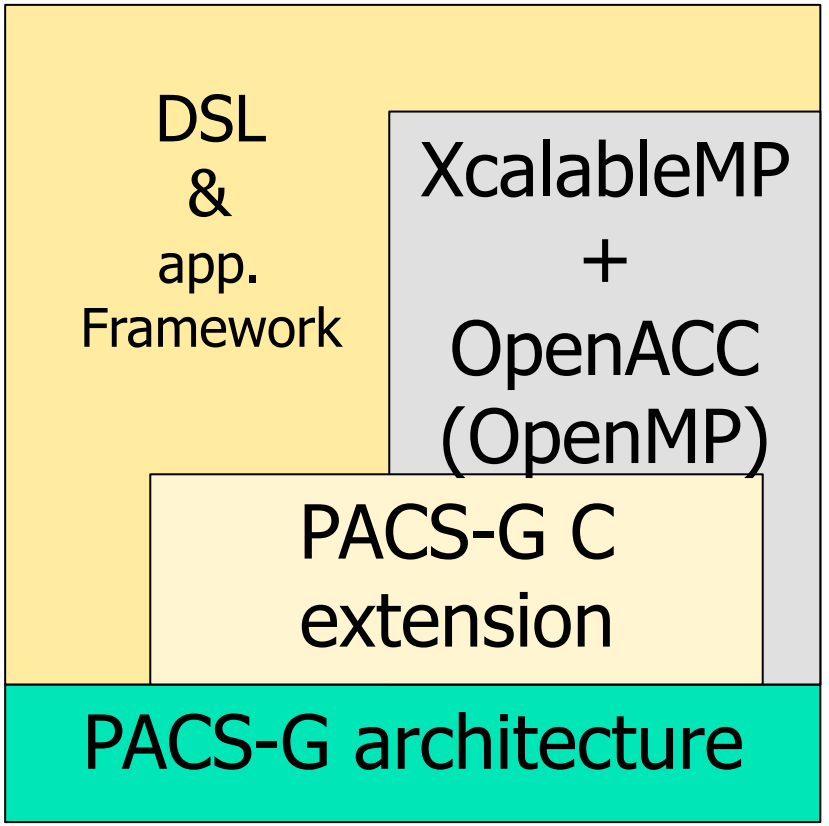
- The size of LM（PE's local memory） is small
- "extreme" SIMD (all PE execute the same instruction)
- No random access to GM, only block transfer
  - can be overlapped with computation in PE

- support for porting existing software
  - directive-based extension

- Message passing model may be used between processors
  - MIMD

Programming models for PACS-G

- PACS-G C extension for low-level programing
  - low-level constructs to program PE, and assembly level program by Intrinsic's
- XcalableMP (subset/extension) + OpenACC for directive based programming for stencil apps.
  - to make it easy to port existing codes (stencil codes)
- Domain-Specific Language (DSL) and app. framework for more optimized code.
  - ex. DSL for stencil or/and particle simulation
- Programing support to describe offloaded code from host by directives
  - Compiled to low-level host interface
- For inter-chip network, MPI or proprietary communication lib will be supported.



PACS-G C extension

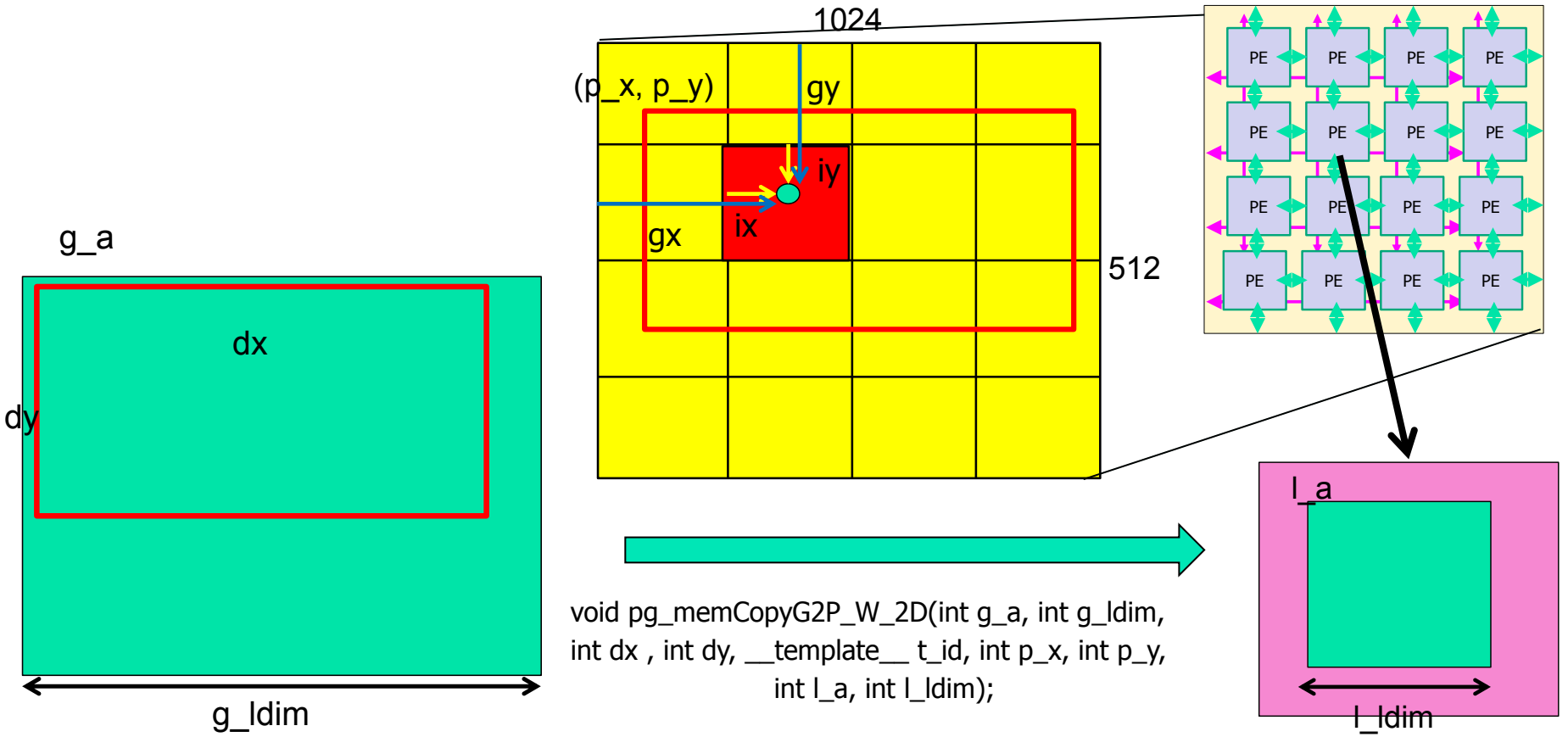
XcalableMP

- 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



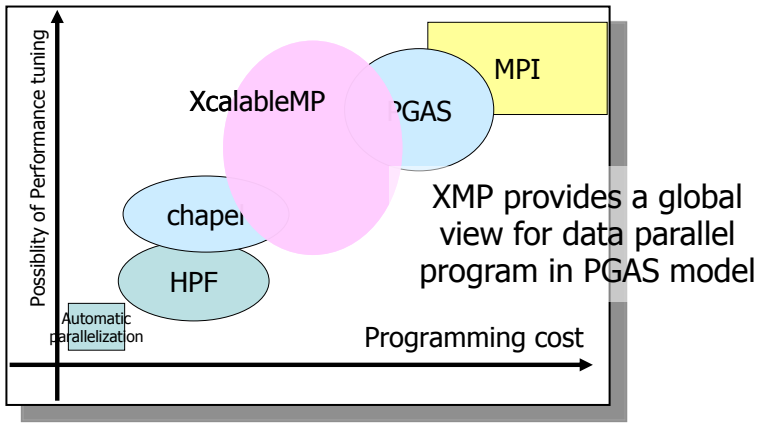
XcalableMP+OpenACC (or OpenMP)

- Optimization using LM（PE's local memory） explicitly by controlling and allocating data in LM ⇒ XcalableMP PACS-G extension
  - When all data fits in the size of LM
  - Reflect operation to communicate shadow data between PE's explicitly.

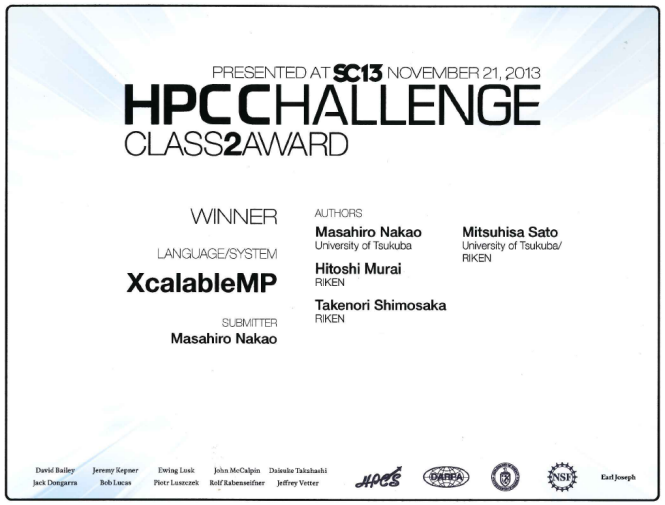
- Using LM as temporary memory for data in GM
  - In case that all data cannot fit in LM
  - Compiler must generate code to overlap computations and data transfer GM⇒LM, LM⇒GM
  - User may indicate parallel loop by directive/pragma (like OpenMP)

XcalableMP(XMP) http://www.xcalablemp.org

- What's XcalableMP (XMP for short)?
  - A PGAS programming model and language for distributed memory , proposed by **XMP Spec WG**
  - XMP Spec WG is a special interest group to design and draft the specification of XcalableMP language. It is now organized under **PC Cluster Consortium**, Japan. Mainly active in Japan, but open for everybody.
- Project status (as of Nov. 2013)
  - XMP Spec **Version 1.2** is available at XMP site. new features: mixed OpenMP and OpenACC , libraries for collective communications.
  - Reference implementation by U. Tsukuba and Riken AICS: **Version 0.7 (C and Fortran90)** is available for PC clusters, Cray XT and K computer. Source-to- Source compiler to code with the runtime on top of MPI and GasNet.



- Language Features
  - Directive-based language extensions for Fortran and C for PGAS model
  - Global view programming with global-view distributed data structures for data parallelism
    - SPMD execution model as MPI
    - pragmas for data distribution of global array.
    - Work mapping constructs 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).



An example of XMP/G Fortran (Laplace)

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

define template (index space over PE) to align array and loop

copy from GM to LM (OpenACC)

Specify distribution on LM

Define shadow area of data on LM

Start parallel region to execute PE(OpenACC)

update shadow area

parallel loop on template

Stencil Computation

End of eecuteion on PE (OpenACC)

Copy back from LM to GM (OpenACC)

Performance Study of PACS-G (1)

- We assume the following two types of configurations for performance estimation
  - Type A： the number of PE 4096. The size of LM is 64KB. To improve power-efficiency and meet the limitation of power, the clock is 750MHz.
  - Type B: the number of PE is 2048 and the size of LM is 128KB. To increase performance, the clock is 1GHz.
- The number of processor in a group is 4096.
- Performance is estimated by picking up the kernel and programming at assembly level
- High-level programming and optimization is not ready yet.

parameter	type A	type B
PE/chip	4096	2048
clock (GHz)	0.75	1
#FLOP(double)/cycle/PE	4	4
Peak FLOPS/chip (TF)	12.3	8.2
LM size(KB)	64	128
BW/PE (GB/s)	12	16
B/F	4	4
On-chip network BW(GB/s)*1	6	8
GM size（GB)	16	16
BW/chip (GB/s)	1024	512
B/F	0.08	0.06
Inter-chip network BW(GB/s) *2	20	20
Chip/Group	4096	4096
Peak Flops/group (PF)	50.3	33.5
LM/group (PB)	1	1
GM/group (PB)	64	64

1）bandwidth per link (upto 4-dimensional neighbor comm supproted)
2）one-direction bandwidth per link (bi-direction, 4D torus network)

Performance Study of PACS-G （2）

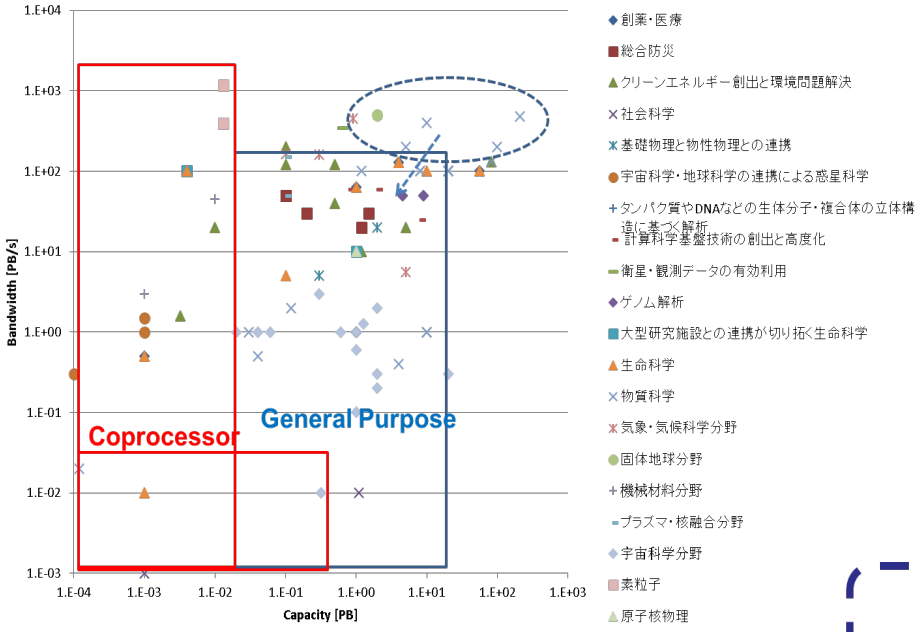
app	typeA	typeB	Comment
Modylas（Molecular Dynamics Simulation）	4.88TF/chip (eff. 39.7%) 0.75ms/step	3.36TF/chip (eff. 41.0%) 1.02ms/step	• The size of molecules is 100M. • Only nearest neighbor force are calculated. Long-distance force are computed in Host. (Offload-model) • Estimated by Assembly level programming for core-kernel • Time for communication to host is 18ms/4step, • Currently, we are considering how to compute long-distance forces in PACS-G
Lattice QCD [CCS-QCD]	Single precision (per processor)		• Offload BiCGStab Solver. Only this part is estimated. • Estimated by Assembly level programming for core-kernel. The time of communication are also estimated. • Using 2048 processors for both of typeA, typeB • The time of single precision and double precision are 90% and 10% respectively. By this ratio, we estimate the total performance. • The time of reduction between processors are estimated 3.5μsec
	5.9TF (eff. 24.0%)	4.60TF (eff. 28%)	
	Double precision (per processor)		
	1.75TF (eff. 14.2%)	0.98TF (eff. 12.0%)	
	Total 5.32TF	total 3.55TF	
Seism3D (Sesimic Simlaute)	Size 2048x2048x1536 (on LM)		• Typical Stencil code • Estimated by Assembly level programming for core-kernel. The time of communication are also estimate • Using one group. • Need overlap of communication between computation and data transfer to/from GM
	18.2PF (eff. 18.1%) 78.4μsec/step	12.9PF (eff. 19.2%), 115.5μsec/step	
	Size 10240x10240x6144(on GM)		
	4.87PF (eff. 4..83%)	2.58PF (eff. 3.84%)	

Summary of results

- Results of MD simulation "Modylas" shows that one time step of 100M molecules may be performed in order of milli-second in real-time by strong scaling.
  - We hope it will accelerate researches of life science such as protein folding.
  - In current version of code, only near-distance interaction is computed by offloading to accelerator. Currently, we are investigating a method to compute long-distance forces.
- Results of latticed QCD shows that the computation can be significantly speedup by using on-chip memory and network, and low-latency accelerator network.
- Stencil computations, which is typical type of scientific applications, were evaluated.
  - If data fits in the size of LM, significant speedup may be obtained.
  - When data is stored in GM, the performance may be restricted by the bandwidth between chip and GM. Even in this case, explicit data transfer may be better than cache, with better power-performance efficiency.
- Results of Seism3D shows that:
  - For small size problem（2048×2048×1536）which fits in LM, one step can be computed in about 0.1msec. It means simulation of small area (e.g, Kanto area) can be done faster than in real-time. It may also be useful to speedup ensemble simulations to execute many cases.
  - For larger size which must be stored in GM, simulation of certain size (e.g. Japan) can be done in a few hours.

(Yutaka Ishikawa @ BDEC2014, Fukuoka)

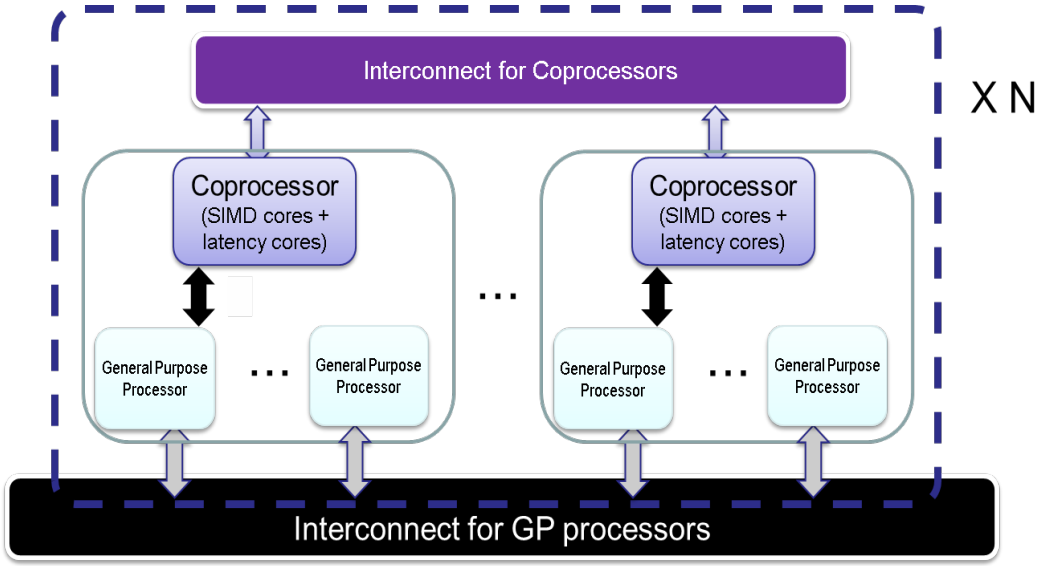
Strawman of postK (Proposed by RIKEN)



Coprocessor
- SIMD engines with latency core
- Strong scale network

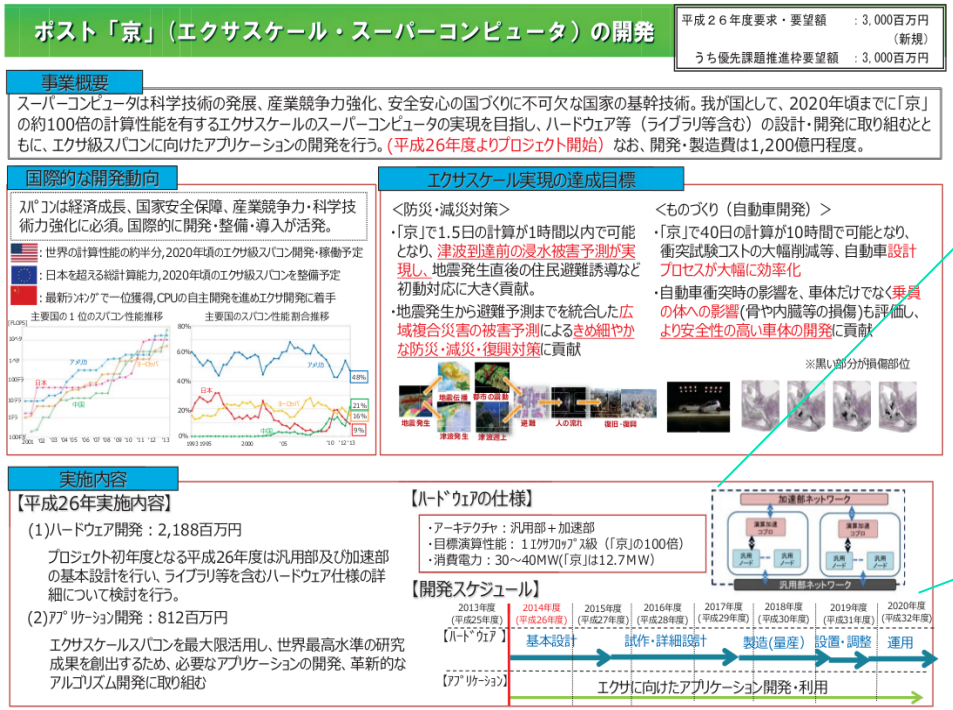
General purpose processor
- Many cores with SIMD instructions
- Tofu network

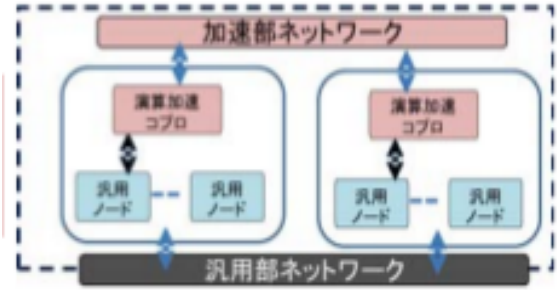
Execution Models
- Separation
- Cooperation
- Offload



Current Status for "exascale" project

- Other software components, low-level communication, MPI implementation, and file I/O have been studied
- This feasibility study is not directly taken over in Japan exascale project being proposed, but a part of this study will be taken over in the project
  - Expected operation year is 2020





http://www.mext.go.jp/component/b_menu/other/_icsFiles/afieldfile/2013/08/30/1339148_5.pdf

Concluding remarks

- To realize "exascale" (≒exaflops), dedicated architecture of accelerator will be required (Power-performance efficiency > 50GF/W, It may be difficult for general-purpose processors)
- For new computational science, we need architecture for strong scaling.

- PACS-G architecture offers high performance and high power-performance efficiency by "extreme SIMD" and on-chip memory/on-chip network, strong scaling by dedicated accelerator network. and SIMD.
  - Applications: MD in life sciences, Lattice QCD in particle physics. ...
  - The point is what processors are possible using LSI technology of 2018-2020.

- Agenda remains
  - detail architecture and instruction set design
  - more detail programming model and compiler optimization (communication between PEs and GM)
  - Co-design and more accurate performance evaluation.
  - System software including communication library, IO, FT, McKernel, ...

  - ⇒ Extend To cover wider range of applications.

Concluding Remarks

- PACS-G: Low power (target > 40-50GF/W) & Strong scaling
  - "extreme" SIMD, less power, and easy to synchronize with out overhead
  - Local memory for each PE.
  - block transfer from GM to LM.
  - ... But, less flexibility and difficult to program ...
  - direct network between processors for shorter latency.

- Comparing to GPU
  - power efficiency < 30GF/w ...
  - no direct network
  - ..., but many software is getting matured.

- Comparing to manycore
  - power efficiency < 20GF/W
  - Huge overhead to synchronize million of threads.
  - ..., but many software already exists!!!

Option?

- Ideas for extension
  - Support for random communication between PEs (remote read/write)
  - Support for random memory access between PE⇔GM
  - Muticore/manycore for master processor

	In processor	Between procs
Regular mem access & regular comp. (e.g. stencil)	○	○
Irregular mem access & irregular comp (sparse max)	△（1）	○（2）
Unbalanced comp. & different threads	×	△

△（1） by supporting random comm in PE and random memory access
○（2） by message passing on conventional MPP networks

Thank you for your attention!!!