Challenges for Parallel Programming Models and Languages of post-petascale and extreme scale computing

A Collaboration of AICS for JLESC

Mitsuhisa Sato Team Leader of Architecture Development Team

FLAGSHIP 2020 project

RIKEN Advance Institute of Computational Science (AICS)

22/FEB/2016

Collaborations for JLESC

- **There are 7 topics in the JLESC:**
  - Apps, Mini-apps, Mini-workflow, Benchmarks
  - Resilience
  - I/O Storage and In Situ Processing
  - Programming Languages and runtimes
  - Tools
  - Numerical Methods/algorithms
  - Advanced Architectures (Cloud, FPGA, etc.)

Collaborations for JLESC

- **There are 7 topics in the JLESC:**
  - Apps, Mini-apps, Mini-workflow, Benchmarks
  - Resilience
  - I/O Storage and I
  - Programming Lan
  - Tools
  - Numerical Method
  - Advanced Architectures (Cloud, FPGA, etc.)

AICS Group Leader: Naoya Maruyama
- Fiber Miniapp
- PGAS (Coarray/XMP) version of Fiber (under development)
- Benchmarking methodologies and SPP/SEP (Sustained Petascale/Exascale Performance) Benchmarks (not proposed, under consideration)

Collaborations for JLESC

- **There are 7 topics in the JLESC:**
  - Apps, Mini-apps, Mini-workflow, Benchmarks
  - Resilience
  - I/O Storage a Situ Processing
  - Program
  - Tools
  - Numerical Methods/algorithms
  - Advanced Architectures (Cloud, FPGA, etc.)

AICS Group Leader: Atsuhi Hori
- Spare Node Substitution (Presented)
- **“Towards Realistic Fault Resilience”** (Presented)

Collaborations for JLESC

- **There are 7 topics in the JLESC:**
  - Apps, Mini-apps, Mini-workflow, Benchmarks
  - Resilience
  - I/O Storage and In Situ Processing
  - Programming [illegible]ages and runtimes
  - Tools
  - Numerica
  - Advanced

AICS Group Leader: Yutaka Ishikawa

- “Large-Scale Parallel Image Composition for In Situ Visualization Framework” (Kenji Ono’s group, Presented)
- Asynchronous communication with lightweight kernel (to be prposed)

Collaborations for JLESC

- **There are 7 topics in the JLESC:**
  - Apps, Mini-apps, Mini-workflow, Benchmarks
  - Resilience
  - I/O Storage and In Situ Processing
  - Programming Languages and runtimes
  - Tools
  - Numerica
  - Advanced

AICS Group Leader: Mitsuhisa Sato

- XcalableMP Project (reported)
- Collaboration on Omni OpenMP and Argbots (to be proposed), to be extended to multitasking of XMP 2.0

Collaborations for JLESC

- **There are 7 topics in the JLESC:**
  - Apps, Mini-apps, Mini-workflow, Benchmarks
  - Resilience
  - I/O Storage
  - Programming
  - Tools
  - Numerical Methods/algorithms
  - Advanced Architectures (Cloud, FPGA, etc.)

AICS Group Leader: Hitoshi Murai

- Developer tools for porting & tuning parallel appli cations on extreme-scale parallel systems (just sta rted, JSC and BSC)
- Collaboration on VI-HPS (JSC)

Collaborations for JLESC

- **There are 7 topics in the JLESC:**
  - Apps,
  - Resilie
  - I/O St
  - Progra
  - Tools
  - Numerical Methods/algorithms
  - Advanced Architectures (Cloud, FPGA, etc.)

AICS Group Leader: Toshiyuki Imamura

- HPC libraries for solving dense symmetric eigenvalue probl ems (just started, JSC)
- Calculation of eigenvalues and eigenvectors for large spars e non-Hermitian matrices in lattice QCD (to be proposed, J SC)
- Comparison of Meshing and CFD Methods for Accurate Flo w Simulations on HPC systems (just started, JSC)

Collaborations for JLESC

- **There are 7 topics in the JLESC:**
  - Apps, Mini-apps, Mini-workflow, Benchmarks
  - Resilience
  - I/O St
  - Progra
  - Tools
  - Numerical M ds/algorithms
  - Advanced Architectures (Cloud, FPGA, etc.)

AICS Group Leader: Naoya Maruyama

- Future computing technologies with FPGA

Challenges of Programming Languages/models for exascale computing

- Scalability, Locality and scalable Algorithms in system-wide
- Strong Scaling in node
- Workflow and Fault-Resilience
- (Power-aware)

“MPI+X” for exascale?

- **X is OpenMP!**

- **“MPI+Open” is now a standard programming for high-end systems.**
  - I’d like to celebrate that OpenMP became “standard” in HPC programming

- **Questions:**
  - “MPI+OpenMP” is still a main programming model for exa-scale?

Question

- **What happens when executing code using all cores in manycore processors like this ?**

```
 MPI_recv …
#pragma omp parallel for
for ( … ; … ; … ) {
     … computations …
}
 MPI_send …
```

Data comes into "main shared memory"

Cost for "fork" become large

data must be taken from Main memory

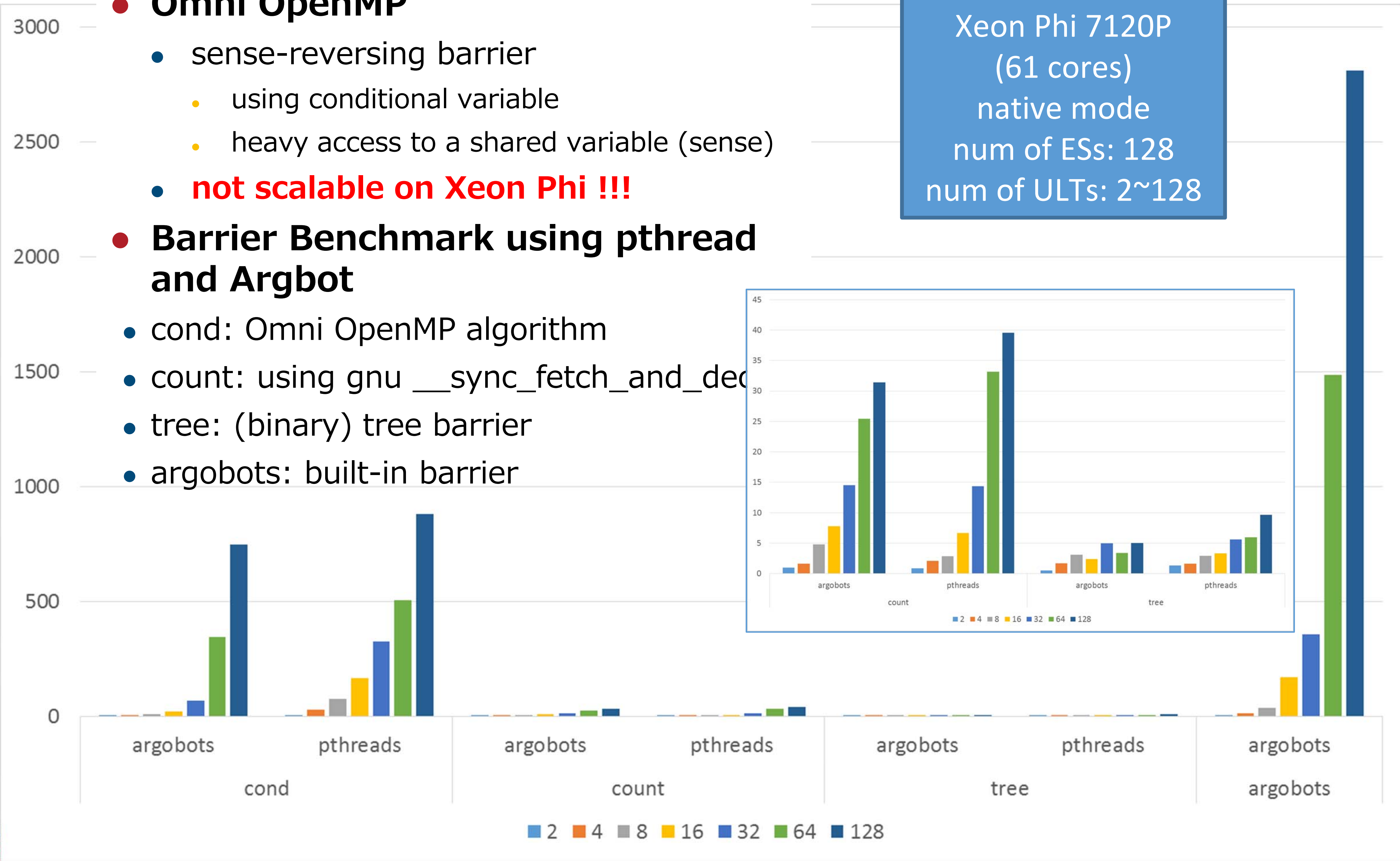
Cost for "barrier" become large

MPI must collect data from each core to send

Barrier in Xeon Phi

- **Omni OpenMP**
  - sense-reversing barrier
    - using conditional variable
    - heavy access to a shared variable (sense)
  - **not scalable on Xeon Phi !!!**
- **Barrier Benchmark using pthread and Argbot**
  - cond: Omni OpenMP algorithm
  - count: using gnu __sync_fetch_and_dec
  - tree: (binary) tree barrier
  - argobots: built-in barrier

Xeon Phi 7120P
(61 cores)
native mode
num of ESs: 128
num of ULTs: 2~128



Question

- **What happens when executing code using all cores in manycore processors like this ?**

```
 MPI_recv …
#pragma omp parallel for
for ( … ; … ; … ) {
     … computations …
}
 MPI_send …
```

Data comes into "main shared memory"

Cost for "fork" become large

data must be taken from Main memory

Cost for "barrier" become large

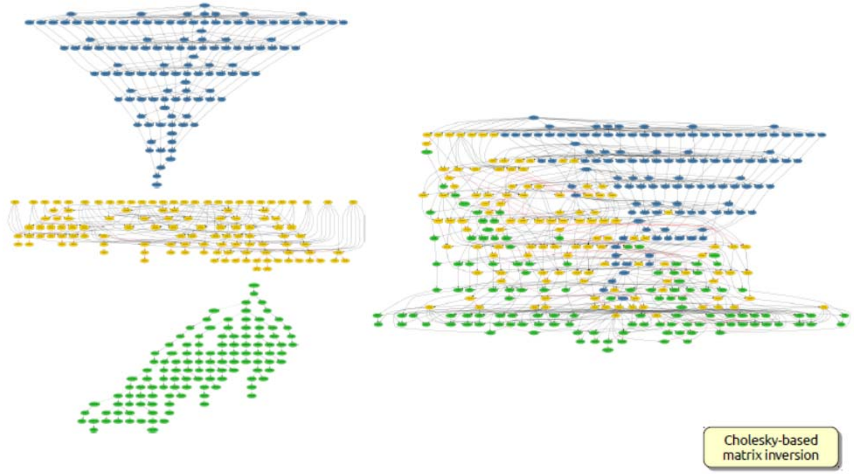
MPI must collect data from each core to send

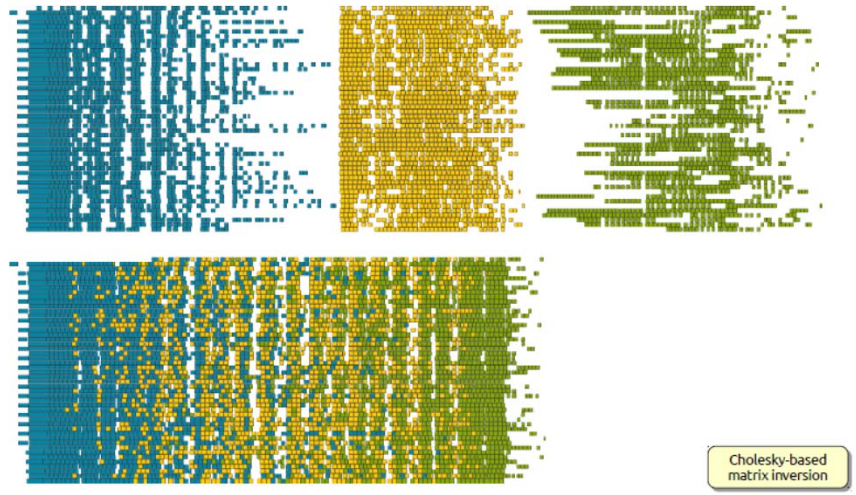
- **What are solutions?**
  - MPI+OpenMP runs on divided small "NUMA domains" rather than all cores?

- Multitasking Models
- PGAS models for Comm.

Multitasking model

- **Multitasking/Multithreaded execution: many "tasks" are generated/executed and communicates with each others by data dependency.**
  - OpenMP task directive, OmpSS, PLASMA/QUARK, StarPU, ..
  - Thread-to-thread synchronization /communications rather than barrier
- **Advantages**
  - Remove barrier which is costly in large scale manycore system.
  - Overlap of computations and computation is done naturally.
  - New communication fabric such as Intel OPA (OmniPath Architecture) may support core-to-core communication that allows data to come to core directly.
- **New algorithms must be designed to use multitasking**

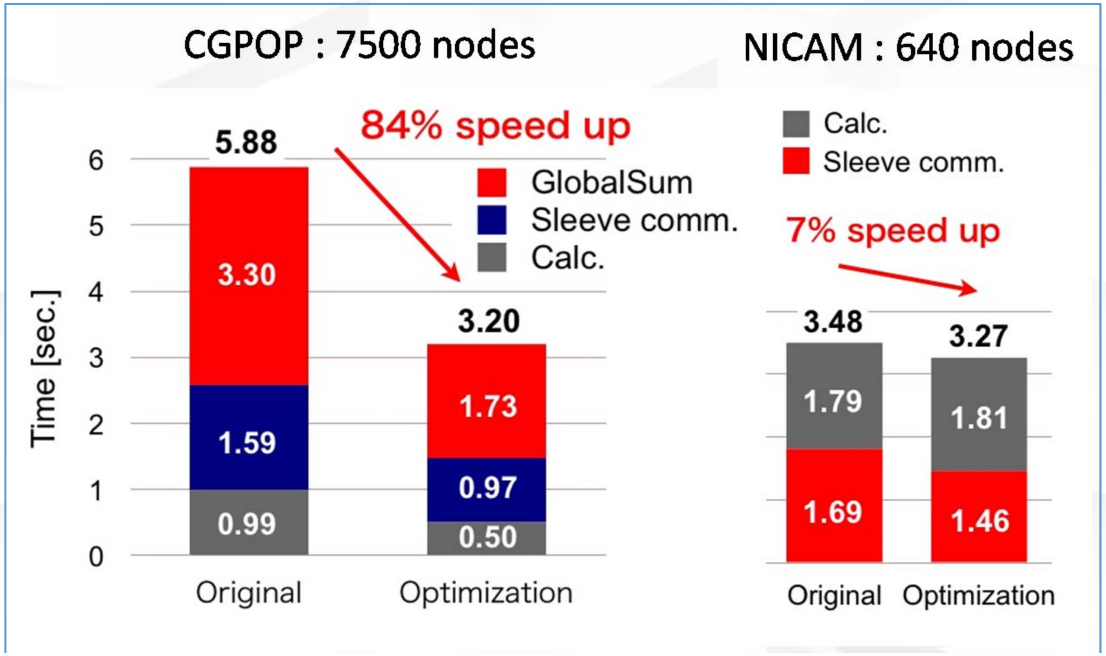




From PLASMA/QUARK slides by ICL, U. Teneessee

PGAS (Partitioned Global Address Space) models

- Light-weight one-sided communication and low overhead synchronization semantics.
- PAGS concept is adopted in Coarray Fortran, UPC, X10, XMP.
  - XMP adopts notion Coarray not only Fortran but also “C”, as “local view” as well as “global view” of data parallelism.
- Advantages and comments
  - Easy and intuitive to describe, not noly one side-comm, but also strided comm.
  - Recent networks such as Cray and Fujitsu Tofu support remote DMA operation which strongly support efficient one-sided communication.
  - Other collective communication library (can be MPI) are required.



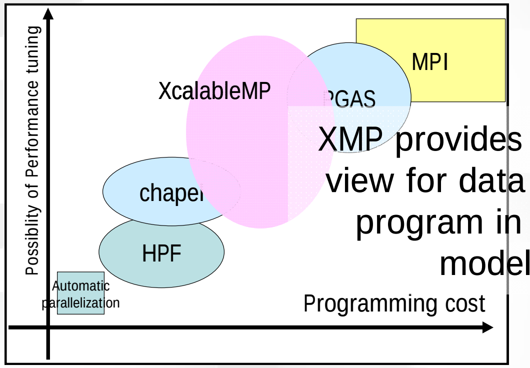
Case study of XMP on K computer
CGPOP, NICAM: Climate code

5-7 % speed up is obtained by replacing MPI with Coarray

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

AICS

- **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. 2014)**
  - 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 0.93 (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.
- **HPCC class 2 Winner 2013. 2014**



XMP provides a global view for data parallel program in PGAS model

- 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).

Code example

```
int array[YMAX][XMAX];

#pragma xmp nodes p(4)
#pragma xmp template t(YMAX)
#pragma xmp distribute t(block) on p
#pragma xmp align array[i][*] to t(i)

main(){
  int i, j, res;
  res = 0;

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

data distribution

add to the serial code : incremental parallelization

work sharing and data synchronization

Omni XcalableMP Compiler

- Production-level Reference impl. being developed by RIKEN & U. Tsukuba.
- Ver. 0.9.2 is available at: `omni-compiler.org`
- Most of XMP features supported.
- Supported Platforms:

  K computer, Fujitsu FX10, Cray, IBM BlueGene, NEC SX, Hitachi SR, Linux clusters, etc.
- Proven applications include:
  - Plasma (3D fluid)
  - Seismic Imaging (3D stencil)
  - Fusion (Particle-in-Cell)
  - etc.

Coarray

- XMP includes the coarray feature imported from Fortran 2008 for the local-view programming.
- Basic idea: data declared as *coarray* can be accessed by remote nodes.
- Coarray in XMP/Fortran is fully compatible with Fortran 2008.

```
real b(8)[*]

if (xmp_node_num() == 1) then
  a(:) = b(:)[2]
```

b is declared as a coarray.

Node 1 *gets* b from node 2.

Coarrays in XMP/C

- Coarrays can be used in XMP/C.
  - The subarray notation is also available as an extension.

- Declaration

```
float b[8]:[*];
```

- Put

```
a[0:3]:[1] = b[3:3];
```

puts b to node 1.

- Get

```
a[0:3] = b[3:3]:[2];
```

gets b from node 2.

- Synchronization

```
void xmp_sync_all(int *status)
```

Note: the syntax of subarray is the same as that in Intel Cilk and OpenACC.
[*start* : *length* : *stride*]

Future Development Plans

- Supporting later vers. of the base languages

  Fortran2008 / C99 / C++11

- Closer cooperation with backend compilers
  - Omni XMP provides them with hints for optimization.
  - etc.
- Alternative libraries for one-sided comms.

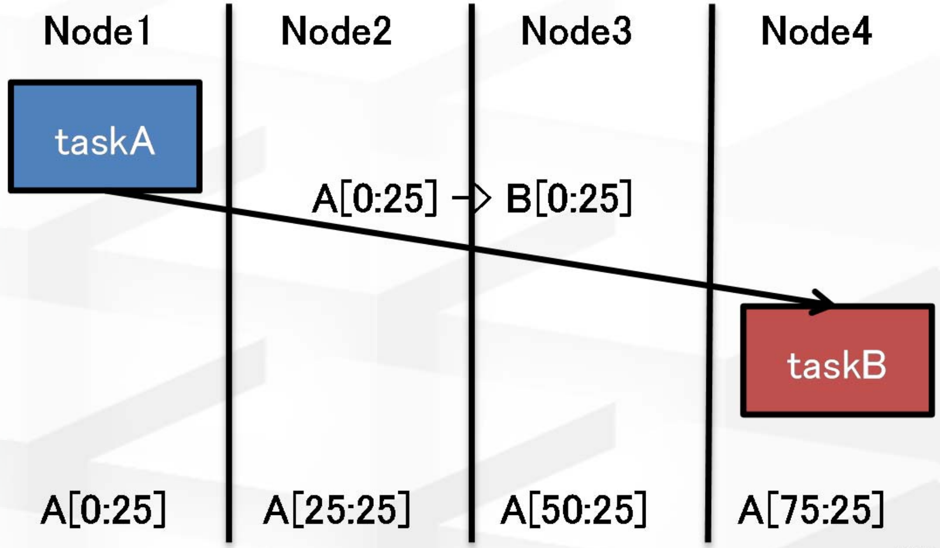
  GASPI, OpenSHMEM, etc.

- Supporting the new features.
- Tool I/F

XcalableMP 2.0

- **Specification v 1.2:**
  - Support for Multicore: hybrid XMP and OpenMP is defined.
  - Dynamic allocation of distributed array
- **A set of specs in version 1 are now "converged". New functions should be discussed for the next version.**
- **Main topics for XcalableMP 2.0: Support for manycore**
  - Multitasking with integrations of PGAS model
  - Synchronization models for dataflow/multitasking executions
  - Proposal: tasklet directive
    - Similar to OpenMP task directive
    - Including inter-node communication on PGAS

```
int A[100], B[25];
#pragma xmp nodes P()
#pragma xmp template T(0:99)
#pragma xmp distribute T(block) onto P
#pragma xmp align A[i] with T(i)
/ … /
#pragma xmp tasklet out(A[0:25], T(75:99))
  taskA();
#pragma xmp tasklet in(B, T(0:24)) out(A[75:25])
  taskB();
#pragma xmp taskletwait
```



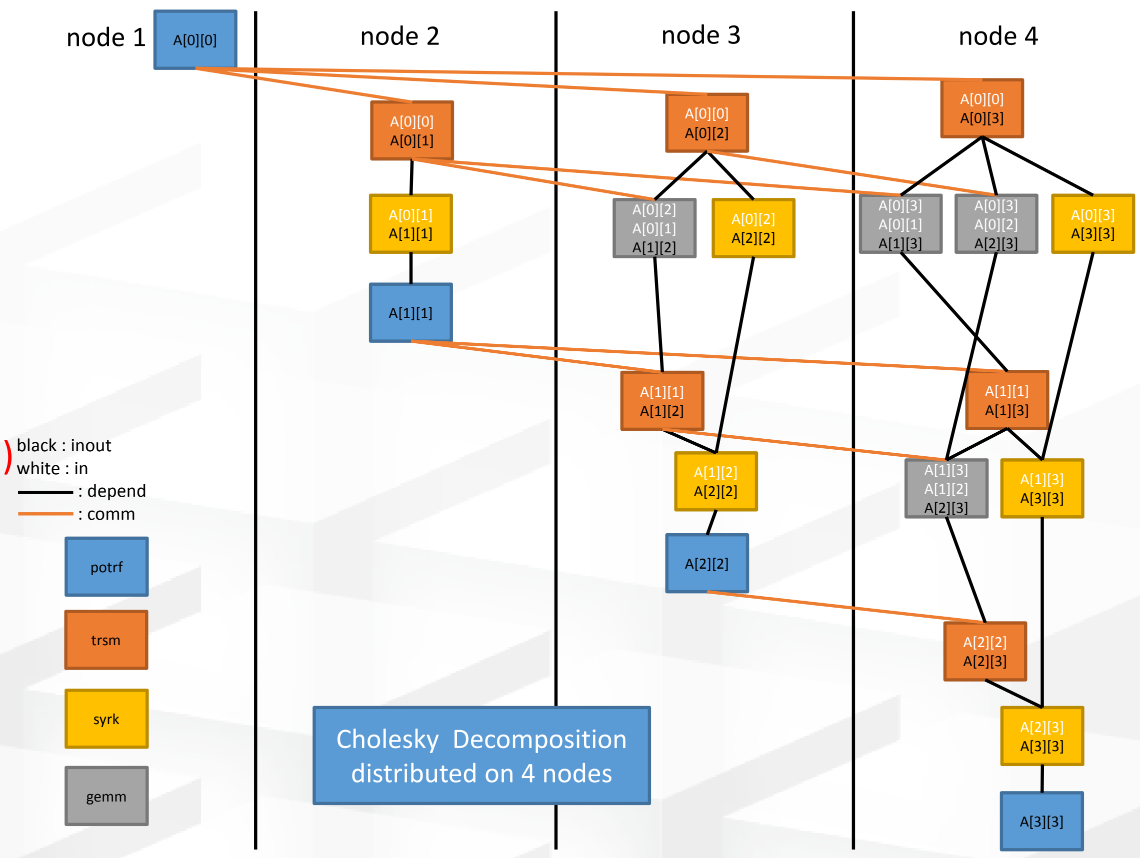
Proposal of Tasklet directive

- **The detail spec of the directive is under discussion in spec-WG**
- **Currently, we are working on prototype implementations and preliminary evaluations**
- **Example: Cholesky Decomposition**

```
double A[nt][nt][ts*ts], B[ts*ts], C[nt][ts*ts];
#pragma xmp node P(*)
#pragma xmp template T(0:nt-1)
#pragma xmp distribute T(cyclic) onto P
#pragma xmp align A[*][i][*] with T(i)

for (int k = 0; k < nt; k++) {
  #pragma xmp tasklet inout(A[k][k], T(k+1:nt-1))
  omp_potrf (A[k][k], ts, ts);

  for (int i = k + 1; i < nt; i++) {
    #pragma xmp tasklet in(B, T(k)) inout(A[k][i], T(i+1:nt-1))
    omp_trsm (B, A[k][i], ts, ts);
  }
  for (int i = k + 1; i < nt; i++) {
    for (int j = k + 1; j < i; j++) {
      #pragma xmp tasklet in(A[k][i]) in(C[j], T(j)) inout(A[j][i])
      omp_gemm (A[k][i], C[j], A[j][i], ts, ts);
    }
    #pragma xmp tasklet in(A[k][i]) inout(A[i][i])
    omp_syrk (A[k][i], A[i][i], ts, ts);
  }
}
#pragma xmp taskletwait
```



Strong Scaling in node

- **Two approaches:**
  - SIMD for core in manycore processors
  - Accelerator such as GPUs

- **Programming for SIMD**
  - Vectorization by directives or automatic compiler technology
  - Limited bandwidth of memory and NoC
  - Complex memory system: Fast-memory (MD-DRAM, HBM, HMC) and DDR , VMRAM

- **Programming for GPUs**
  - Parallelization by OpenACC/OpenMP 4.0. Still immature but getting matured soon
  - Fast memory (HMB) and fast link (NV-Link): similar problem of complex memory system in manycore.
  - Programming model to be shared by manycore and accelerator for high productivity.

Prog. Models for Workflow and data managements

- Petascale system was targeting some of “capability” computing.
- In exascale system, it become important to execute huge number of medium-grain jobs for parameter-search type applications.

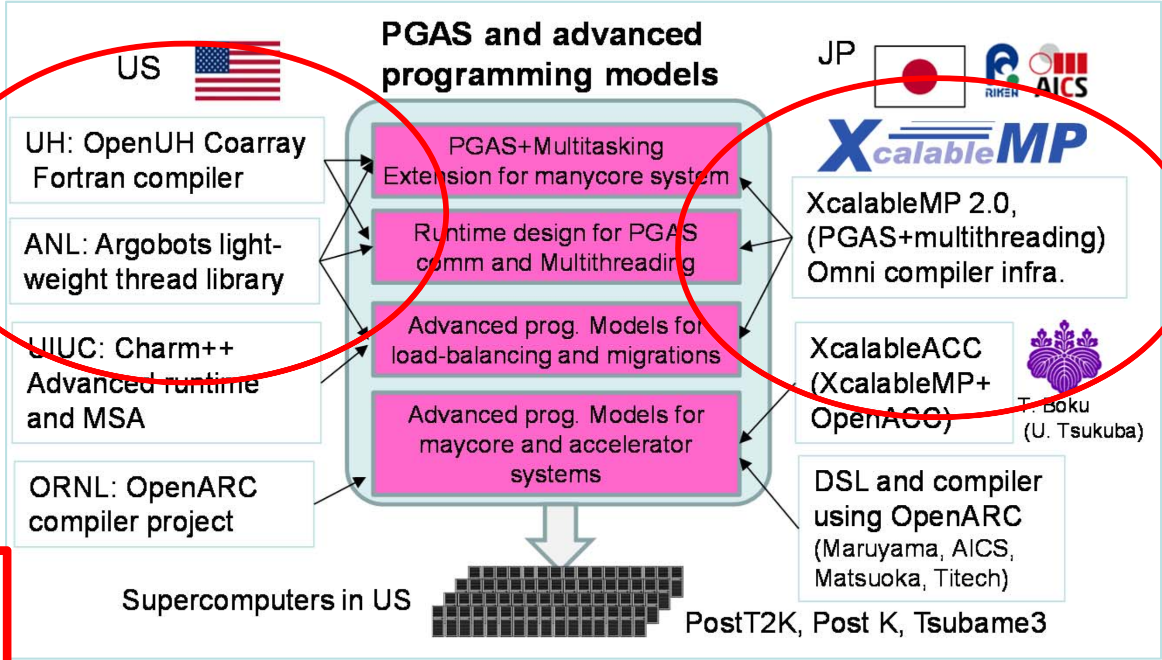
Workflow to control and collect/process data is important, also for “big-data” apps.

International Collaborations

- **JLESC project: VI-HPS (Virtual Institute - High Productivity Supercomputing)**
  - Design of Performance tools interface for XMP and XMP 2.0
  - XMP tutorial
- **DOE-MEXT US-JP collaborations on system software**
  - Use Argobots (light-weight user-level thread lib) for multitasking and OpenMP impl.
  - Interface to Charm++
- **SPPEXA-2**
  - Collaborations with RWTH Aachen, U. Tsukuba and AICS, MDLS

PGAS and Advanced programming models for exascale systems

- Coordinators
  - US: P. Beckman (ANL), JP: M. Sato (RIKEN)
- Leaders
  - US: L. Kale (UIUC), B Chapman (U Huston), J. Vetter (ORNL), P. Balaji (ANL)
  - JP: M Sato (RIKEN)
- Collaborators
  - S. Seo (ANL), D Bernholdt (ORNL), D. Eachempati(UH)
  - H. Murai (RIKEN), J. Lee (RIKEN), N. Maruyama (RIKEN), T. Boku (U. Tsukuba)
- Collaboration topics
  - Extension of PGAS (Partitioned Global Address Space) model with language constructs of multitasking (multithreading) for manycore-based exascale systems
  - Runtime design for PGAS communication and multitasking
  - Advanced programming models to support both manycore-based and accelerator-based exascale system for high productivity.
  - Advanced programming models for dynamic load-balancing and migration in exascale systems
- How to collaborate
  - Twice meetings per year
  - Student / young researchers exchange, sharing codes
  - Funding:
    - US: ARGO, X-stack(XPRESS), X-stack(Vancouver, ARES)
    - JP: FLAGSHIP 2020, PP-CREST (JP)



- Deliverables
  - Concepts for PGAS and multithreading integration for manycore-based exascale systems.
  - Concepts for advanced programming model to be shared by both manycore and accelerators-based systems.
  - Pre-standardization of Application Programming Interface for multithreading (based on Argobots) and PGAS
- Recent activities and plans
  - AICS teams visited UH, UIUC and ANL for discussions.
  - Start using Argobots for Omni OpenMP compiler and produced preliminary results on intel Xeon Phi.
  - AICS invited Post-doc from UH for collaborations on PGAS
  - ORNL visited AICS to have a meeting for the collaboration
  - JP (AICS , Tsukuba) will send Post-doc and students to ANL and UH, ORNL
  - JP and ORNL will have a meeting in JP or US how to collaborate.

Final remarks on JLEC activity in AICS

- **We are pleased to work on the collaborations with international partners in JLESC.**
- **Especially, we are very interested in the performance evaluation and benchmarking for very high-end supercomputers.**
  - Our government and communities want any measures and evaluation methods in term of ROIs of supercomputing …
- **Promotion of collaborations on "applications" and computational sciences.**