Software Researches for Big Data and Extreme-Scale Computing

System Software for Post-Peta Scale Computing / Extreme Big Data

Research and Development of this section is supported by:

- Development of System Software Technologies for post-Peta Scale High Performance Computing (www.hpcs.cs.tsukuba.ac.jp/project/crest-ppfs/en)
- EBD: Extreme Big Data – Convergence of Big Data and HPC for Yottabyte Processing (www.extreme-bigdata.jp)
- Statistical Computational Cosmology with Big Astronomical Imaging Data

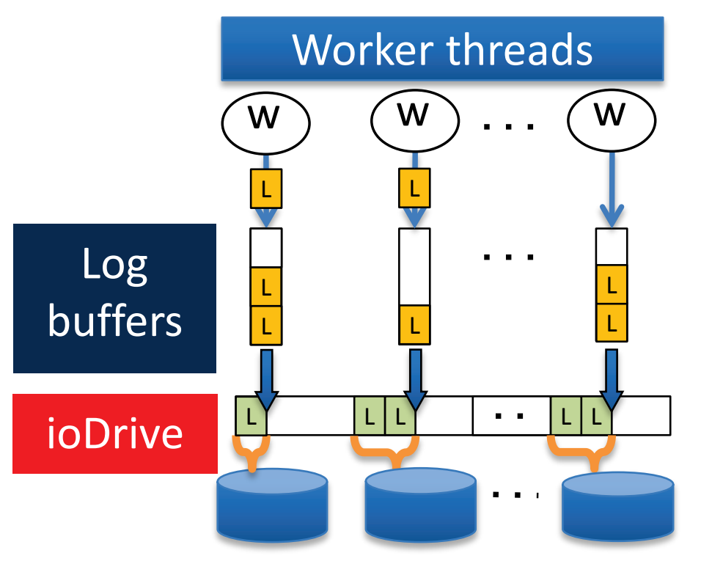
JST CREST Japan Science and Technology Agency

P-WAL: Parallel Write Ahead Logging

P-WAL is a new WAL protocol designed for flash storage, especially ioDrive.
P-WAL can cope with write-ahead logging overheads such as serialization and storage I/O.
P-WAL's features are as follows:

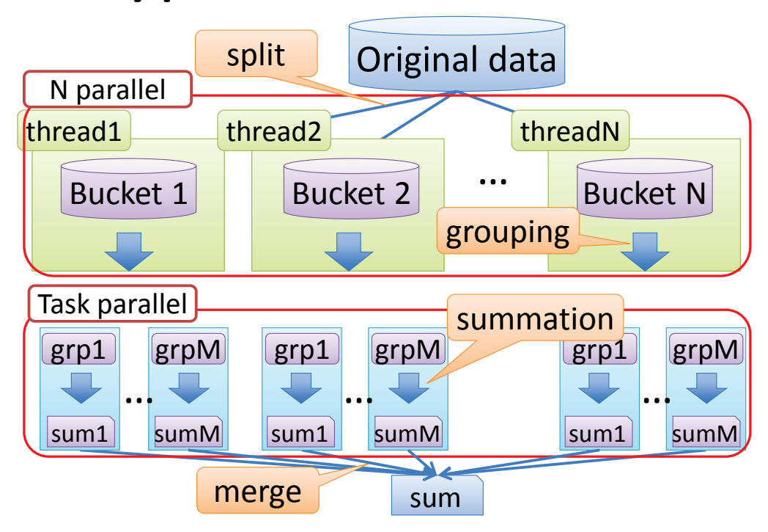
- **Having no shared log buffers**
- **Multiplexing storage I/O**

(SOSP 2015 Poster)



Efficient Summation on Encrypted DBMS

- Existing encrypted DBMS is inferior in performance due to encryption.
- We research highly efficient query processing on encrypted DBMS.
  – We propose parallel summation scheme that fits encrypted DBMS. It is 5.86 times faster than existing schemes.



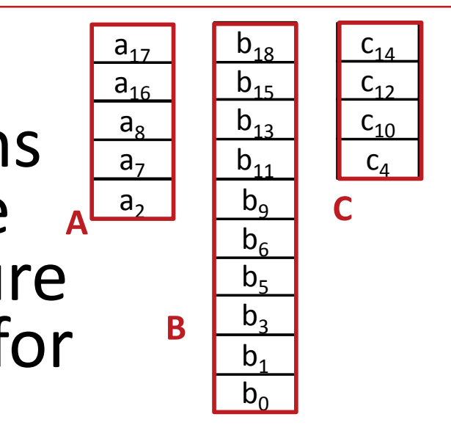
Fast Complex Event Processing over Streams

Input: b_0 b_1 a_2 b_3 c_4 b_5 b_6 a_7 a_8 b_9 c_{10} b_{11} c_{12} b_{13} c_{14} b_{15} a_{16} a_{17} b_{18} b_{19}

QUERY
PATTERN **SEQ(A a, B b, C c)**
FROM sequence
RETURN abc

STEP 1: Put events to AIS
STEP 2: Search from C with pruning
STEP 3: Memo events during search
STEP 4: Output events on leaf node

This work accelerates stream data processing. It prunes unnecessary paths in the active instance stacks, stores the information in the concise data structure format, and finally expands in parallel for high efficiency.



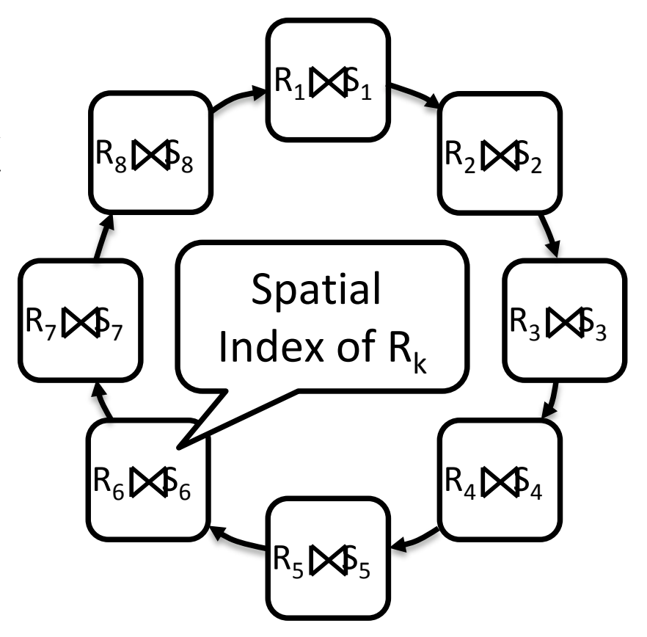
Similarity Search Exploiting Cyclo-Join + Spatial Index

Cyclo-Join is a distributed join technique that circulates one table S. To speed up spatial join, **Spatial Index** are commonly used. We mix these join techniques for faster index construction & similarity search.

Applications

- Star catalogue cross-matching
- Guerrilla rainstorm prediction

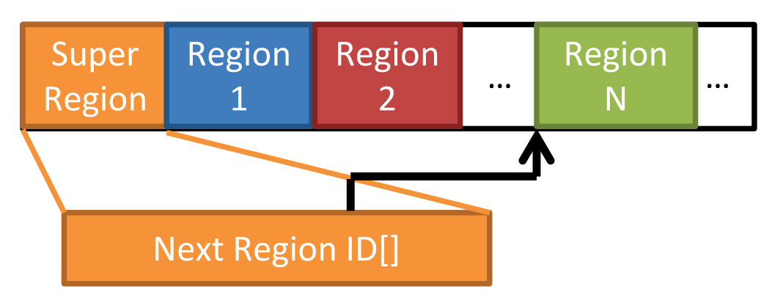
With 10^6 2D-points, our experiment shows x26.5 performance than the naïve approach. (30 machines)



Object Storage Design in OpenNVM

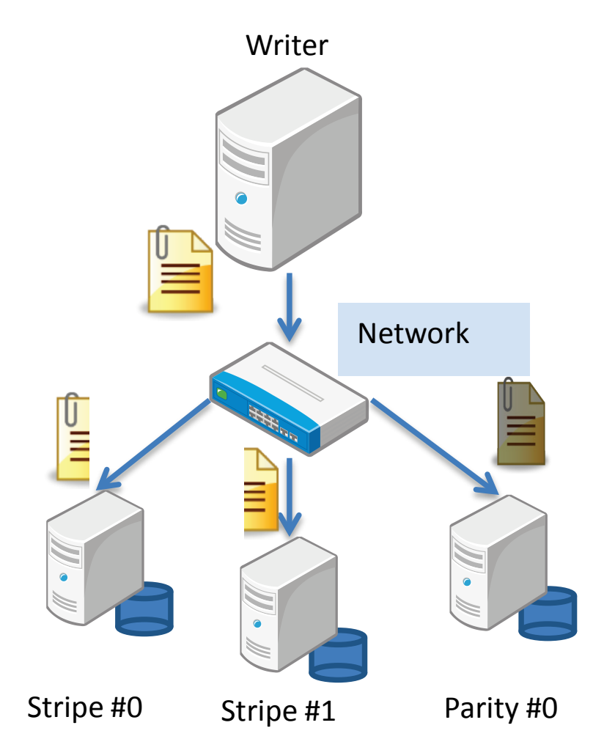
- New interface - Sparse address space, atomic batch operations and persistent trim
- Simple design by fixed-size Region enabled by sparse address space and persistent trim
- Optimization techniques for object creation: Bulk reservation and bulk initialization

(GPC 2015)



Network-based Parity Generation

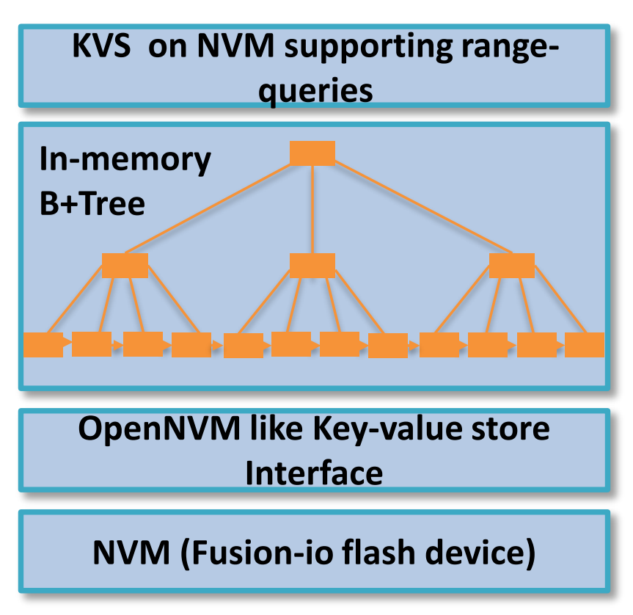
The network-based based parity generation method utilizes programmable abilities of network switches to minimize overheads.
(1) Network-based Data Processing Architecture for Reliable and High-performance Distributed Storage System (BigData Cloud 2015)



NVM-BPTree: Highly Concurrent KVS

NVM-BPTree Highly concurrent Key-Value-Store running natively over Non-Volatile-Memory, supporting range-queries.

- Keys and metadata are stored in a in-memory B+Tree with negligible overhead.
- Support lock-free concurrent Operations. Search queries are not delayed by dynamic rebalancing of the tree.


KVS on NVM supporting range-queries
In-memory B+Tree
OpenNVM like Key-value store Interface
NVM (Fusion-io flash device)

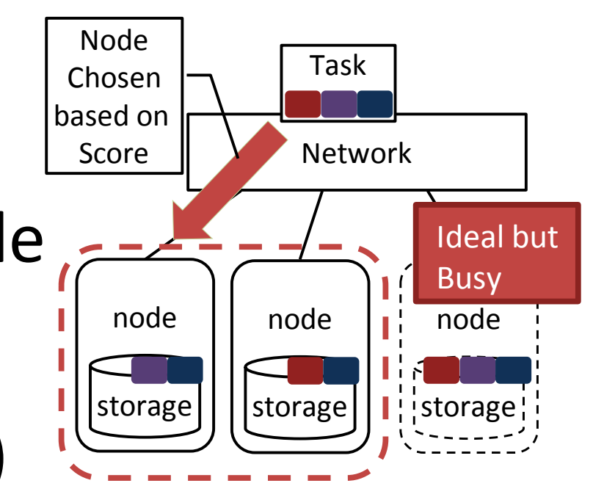
Data-Aware Task Dispatching

- Designed for Batch Queuing System where its underlying file systems' performance is strongly affected by the file allocation.

$$Score = fileLocality \times \beta + load \times (1 - \beta)$$

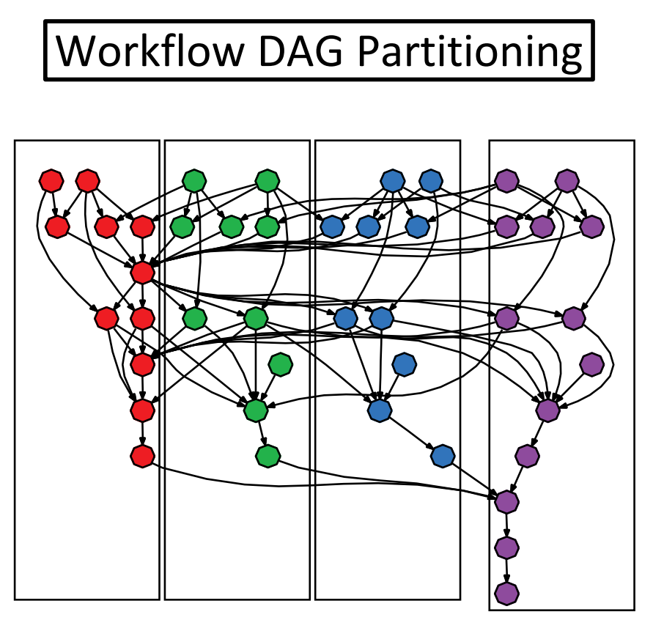
- $\boldsymbol{fileLocality}$: calculated based on file size and placement. ***load*** is the CPU average.

(GPC 2015, IEEE Systems Journal 2015)



Pwrake: Data-intensive Workflow System

Pwrake workflow engine depends on ***Gfarm*** file system for scalable I/O performance. **Task Scheduling** schemes:
(1) **Locality-aware Scheduling** using **Multi-Constraint Graph Partitioning** (CCGrid 2012)
(2) **Disk Cache-aware Scheduling** using LIFO-based task queue (Cluster 2014).



XcalableACC

XcalableACC (XACC) is a PGAS Language for accelerated parallel computers (e.g. GPU clusters), which is a directive-based language extension of C and Fortran

- ➢ High productivity of programming by directives
- ➢ High performance by direct communication between ACCs

Basic Concept

- ●XcalableMP for distributed-memory parallelism

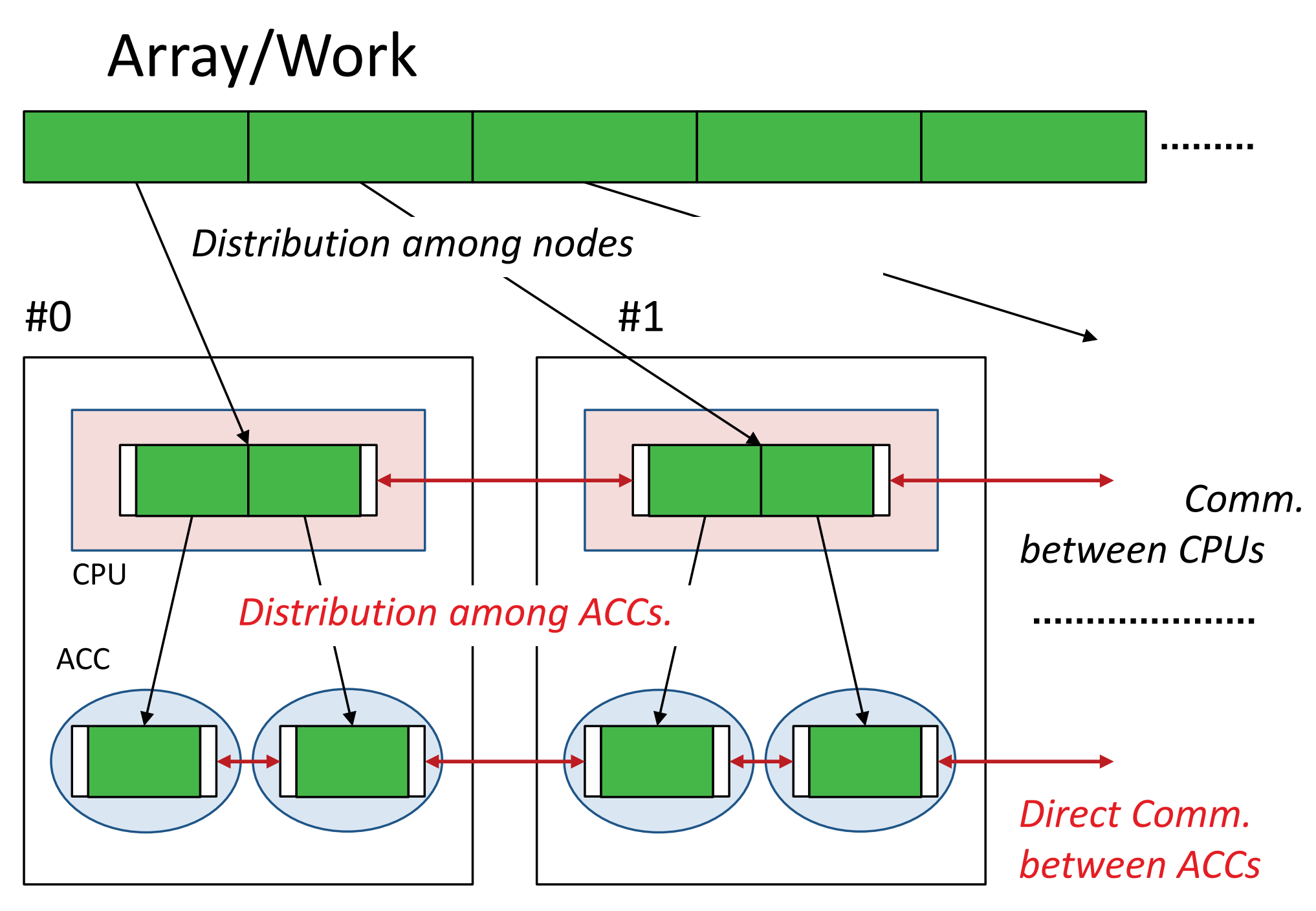
  XcalableMP (XMP) is a directive-based language extension of C and Fortran for distributed-memory parallel systems
- ●OpenACC for offloading works for ACC

  OpenACC is another directive-based language extension for heterogeneous CPU/ACC systems
- ●XACC extensions

XACC Extensions

- ●Data and work mapping onto multiple ACCs (two-level distribution)
  - Distribution among nodes by XMP directives
  - Distribution among ACCs by the novel clauses. (on_device, layout, etc.)
- ●Direct communication between ACCs
  - Communication directives from XMP accept data on the ACC memory.

Data and Work Mapping



HIMENO Benchmark with XACC

```
static float p[MIMAX][MJMAX][MKMAX], …;
#pragma xmp template t(0:MKMAX-1, 0:MJMAX-1, 0:MIMAX-1)
#pragma xmp nodes n(NDZ, NDY, NDX)
#pragma xmp distribute t(block, block, block) onto n
#pragma xmp align p[i][j][k] with t(k, j, i)
#pragma xmp shadow p[1:2][1:2][0:1]
   :
#pragma acc data create(p, …)
{
   :
#pragma xmp loop (k,j,i) on t(k,j,i)
#pragma acc parallel loop reduction(+:gosa) collapse(2)
   for(i=1 ; i<imax-1 ; ++i)
      for(j=1 ; j<jmax-1 ; ++j)
#pragma acc loop reduction(+:gosa) private(s0, ss)
         for(k=1 ; k<kmax-1 ; ++k){
            s0 = p[i+1][j][k] * …;
            ss = … - p[i][j][k] …;
            gosa += ss*ss;
            wrk2[i][j][k] = p[i][j][k] + omega * ss;
         }
   :
#pragma xmp reflect (p) acc
   :
}
```

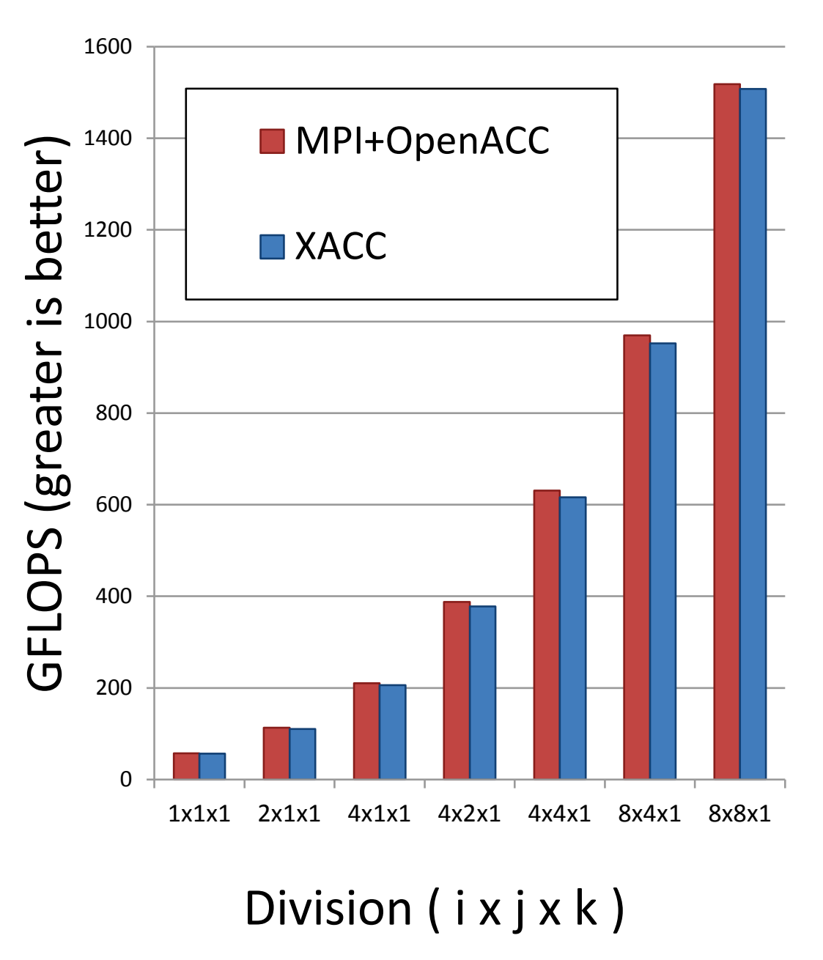
Definition of data mapping

Data allocation on ACC

Loop parallelization and offloading to ACC

Halo comm. on ACC

Performance of HIMENO Benchmark on HA-PACS/TCA



CPU: Xeon-E5 2680v2 x 2
GPU: Tesla K20
Interconnect:
Mellanox Connect-X3 Dual-port QDR
Compiler:
GCC 4.4.7, CUDA 6.5,
Omni XcalableACC compiler
MPI:
MVAPICH2-GDR 2.1a

The performance of the XACC version against MPI+OpenACC version is more than 97%.