

Software Researches for Big Data and Extreme-Scale Computing

System Software for Post Petascale Data Intensive Science
System Software for Extreme Big Data



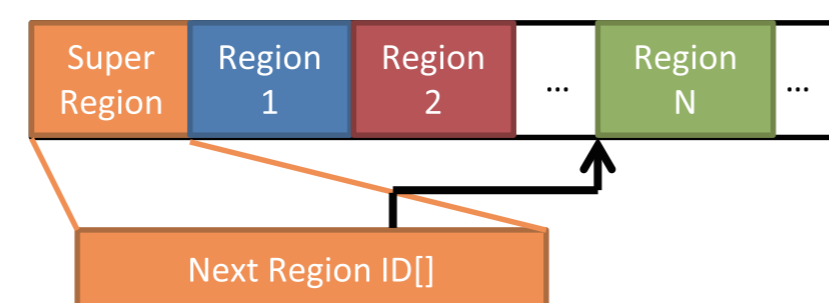
<http://www.hpcs.cs.tsukuba.ac.jp/project/crest-ppfs/en/>
<http://www.extreme-bigdata.jp/>

Distributed File System / Object store

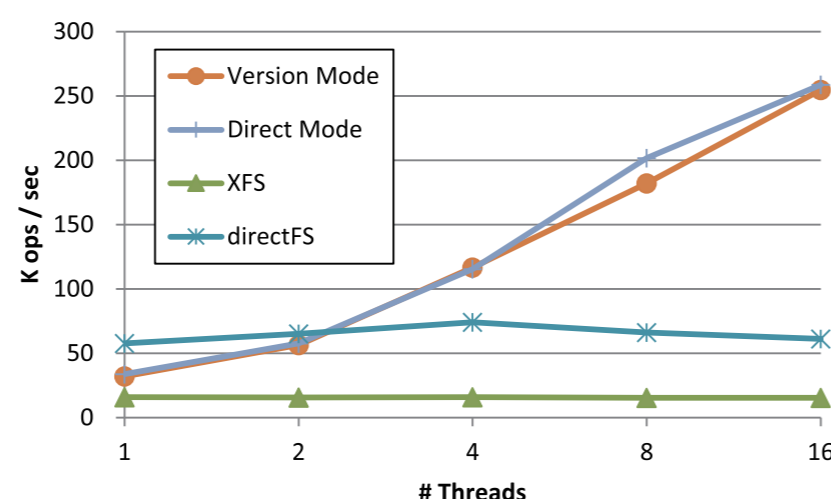
Object Storage for OpenNVM Flash Primitives

Object storage design for high bandwidth and high IOPS in OpenNVM

- OpenNVM flash primitives: **sparse address space and atomic batch write**
- Simple design based on fixed-size **Regions**
 - › One region for one object
 - › Enough region size to store one object
 - › Object ID = region number
- Simple region number management in super region
 - › Sequential assignment
 - › Free'ed by persistent trim and no reuse
 - › Blocked assignment to mitigate access conflict to the super block



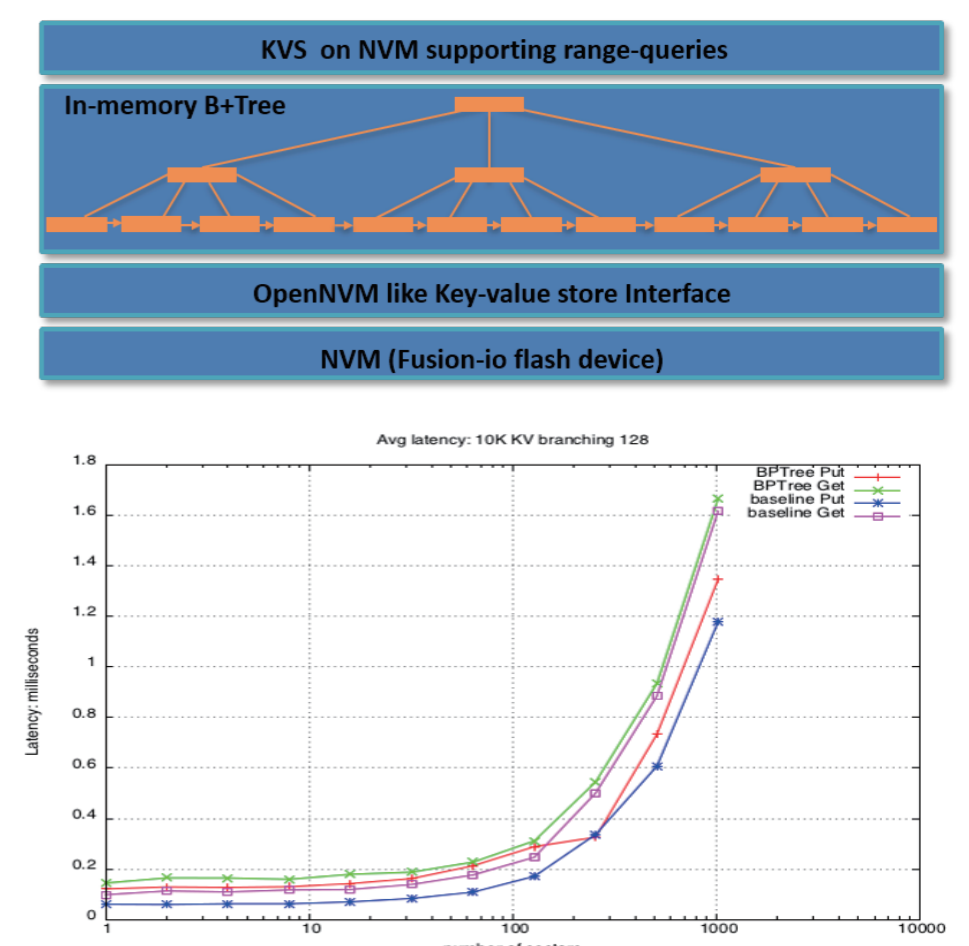
Initial Performance result for object creation



NVM-BPTree

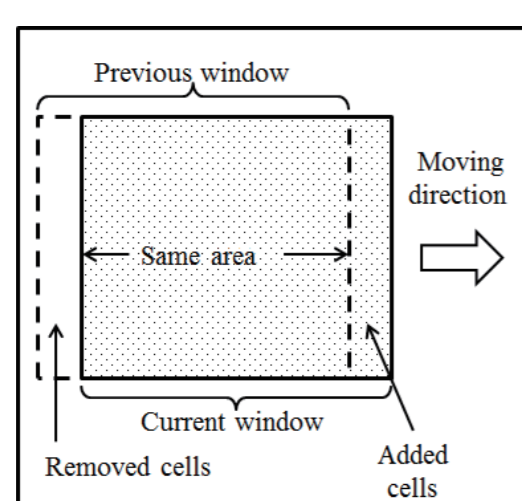
NVM-BPTree is a Key-Value Stores (KVS) running natively over Non-Volatile-Memory (NVM) like flash supporting range-queries.

- Take advantage of enterprise class NVM new capabilities: atomic writes, direct access to NVM device natively as a KVS,...
 - › Leverage *NVMKV* an Open source KVS interface for NVM like flash.
- Enable range-queries support for KVS running natively on NVM
 - › Keys stored in a in-memory B+Tree with negligible overhead for KV pair insertion and retrieval.
- Support lock-free concurrent operation
 - › Search queries are not delayed by the tree's dynamic rebalancing



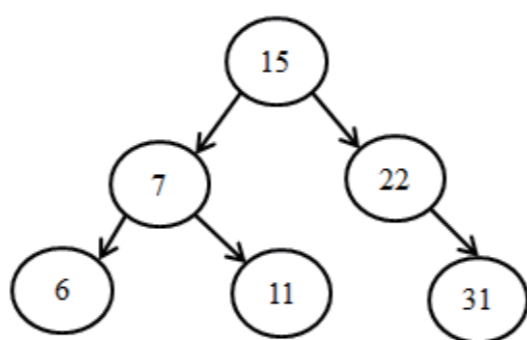
Runtime System

Accelerating percentile with incremental computation scheme over multi-dimensional arrays on SciDB



Basic window

15	7	31	18	2
6	11	22	27	11
31	6	15	9	8
2	9	7	21	13
8	22	7	13	20

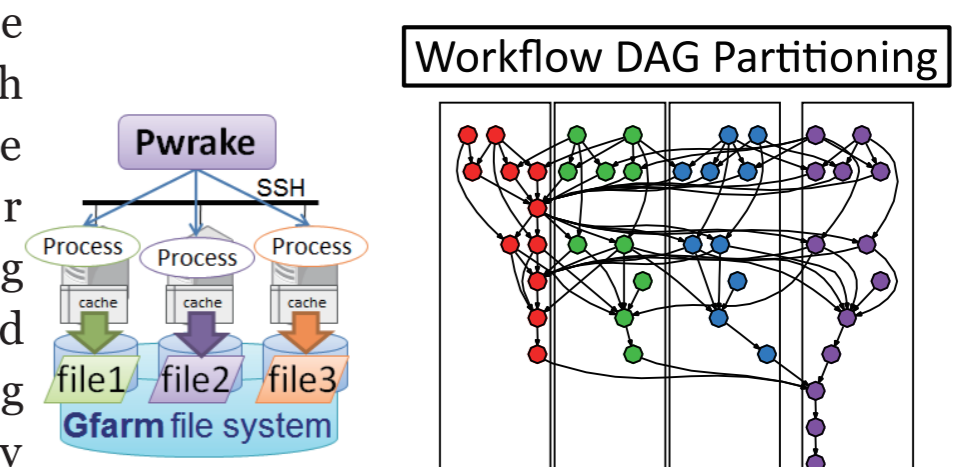


- Keeping values with binary BST
- SciDB implementation shows speed up by a factor of 13

Jiang Li, Hideyuki Kawashima, Osamu Tatebe, "Incremental Aggregates over Array Database", IEEE BigData '14. Codes: <https://github.com/ljiangli/Percentile-in-SciDB.git>

Pwrake: Scalable Workflow System

Pwrake: Workflow engine for data-intensive sciences. For scalable I/O performance, Pwrake depends on **Gfarm** distributed file system which federates local storage of compute nodes. We proposed the following **Task Scheduling** for Pwrake; (1) **Locality-aware Scheduling** using **Multi-Constraint Graph Partitioning** (CCGrid 2012), (2) **Disk Cache-aware Scheduling** using LIFO-based task queue with dynamic priority (Cluster 2014).

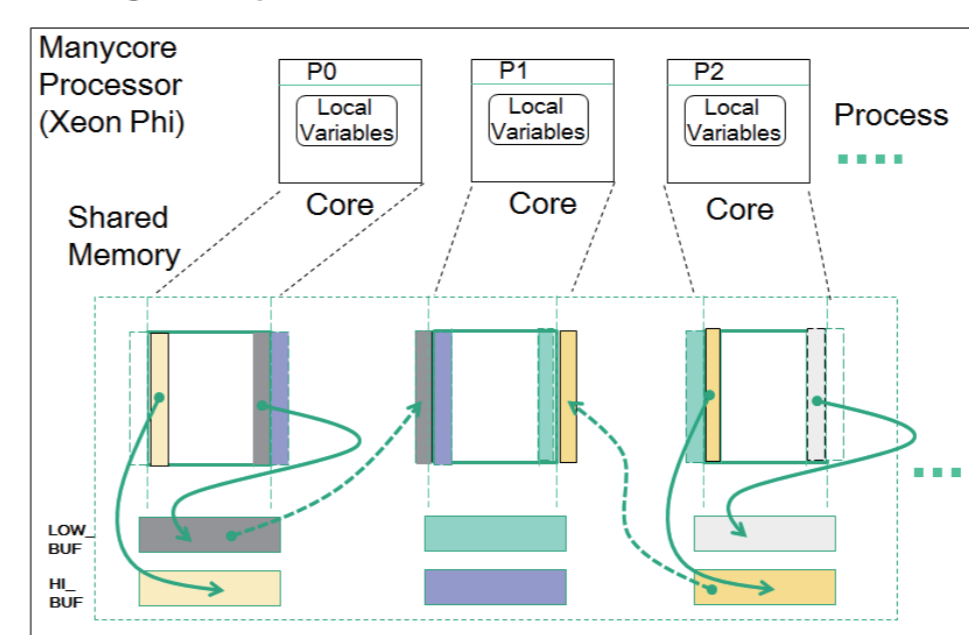


XcalableMP PGAS programming model for Manycore Cluster

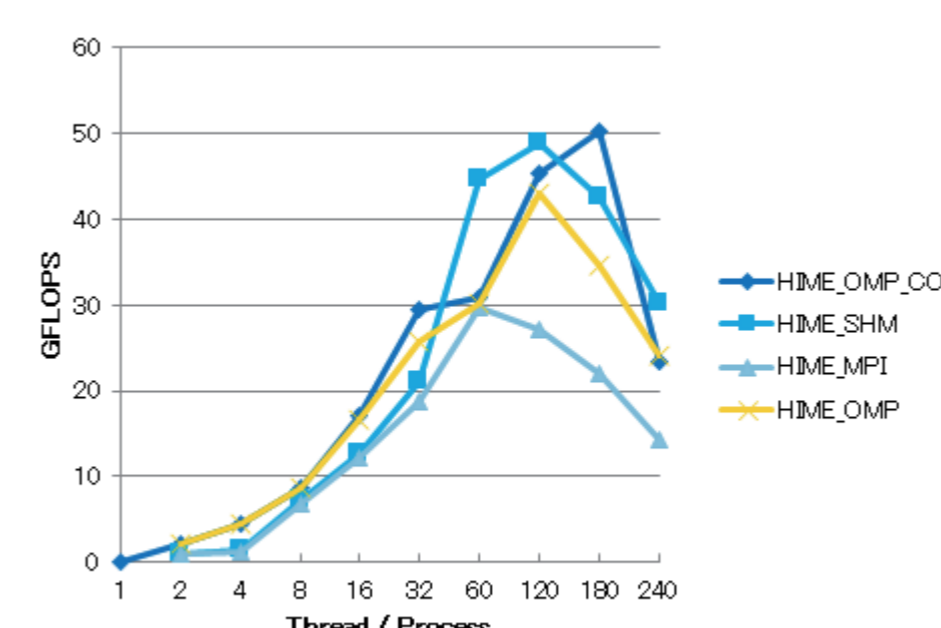
XcalableMP (XMP) is a PGAS language for distributed memory system, which is a directive-based language extension of C and Fortran. Currently, we are carrying out a design & implementation of XcalableMP runtime for Xeon Phi.

- Advantage of XcalableMP PGAS model for Manycore
 - › MPI+OpenMP hybrid is a must in manycore, but its programming cost is high.
 - › XcalableMP provides a unified programming model for both inter-node and intra-node parallelism. The runtime can be designed to make use of message passing for inter-node and shared memory for intra-node for efficient communication.
 - › PGAS model allows the programmers to exploit locality in each core while providing a global address space.
- Design alternatives of XcalableMP for Manycore
 - (1) Using MPI even for intra-node (as same as for inter-node)
 - (2) Using shared memory allocated by mmap, shared by thread for intra-communication. (← the result shown in this poster)
 - (3) Program transformation from PGAS model (based on process) to thread model.
 - (4) Using a new thread model, PVAS (Partitioned Virtual Address Space) supported by Riken AICS, system software team.

Ghost region update on SHM in XMP runtime



Results of Himeno benchmark



Reference: Mitsuru Ikeji, Mitsuhsa Sato, "A PGAS Execution Model for Efficient Stencil Computation on Many-Core Processors." CCGRID 2014: 305-314

Results: Our overall performance advantage against OpenMP is 16.8 % for Lap1D and 13.8 % for Hime2D.

- ① **Blocking Effect of Global Arrays**: By dividing global arrays into sub-arrays detached each others, we could reduce 22.2 % and 27.3 % of CPU calculation time on Laplace and Himeno respectively.
- ② **Efficient Ghost Region Exchange**: By changing MPI sendrcvs to direct memory copy on SHM, we could reduce memory traffics about 32.3% to 44.6% and get the maximum performance gain 62.9 %.