Parallel Access of Dense Extendible Arrays

Ekow Otoo and Doron Rotem

Lawrence Berkeley National Laboratory

Outline

Problem Statement

Problem:
An allocation of a multidimensional array in a parallel file system such that:

- any dimension is allowed to expand.
- parallel applications read/write/manipulate entire array or sub-arrays
- array can be extended without reorganizing previously allocated elements,
- define a mapping function and its inverse for element access.

Data types: integers, floats, double and complex types.

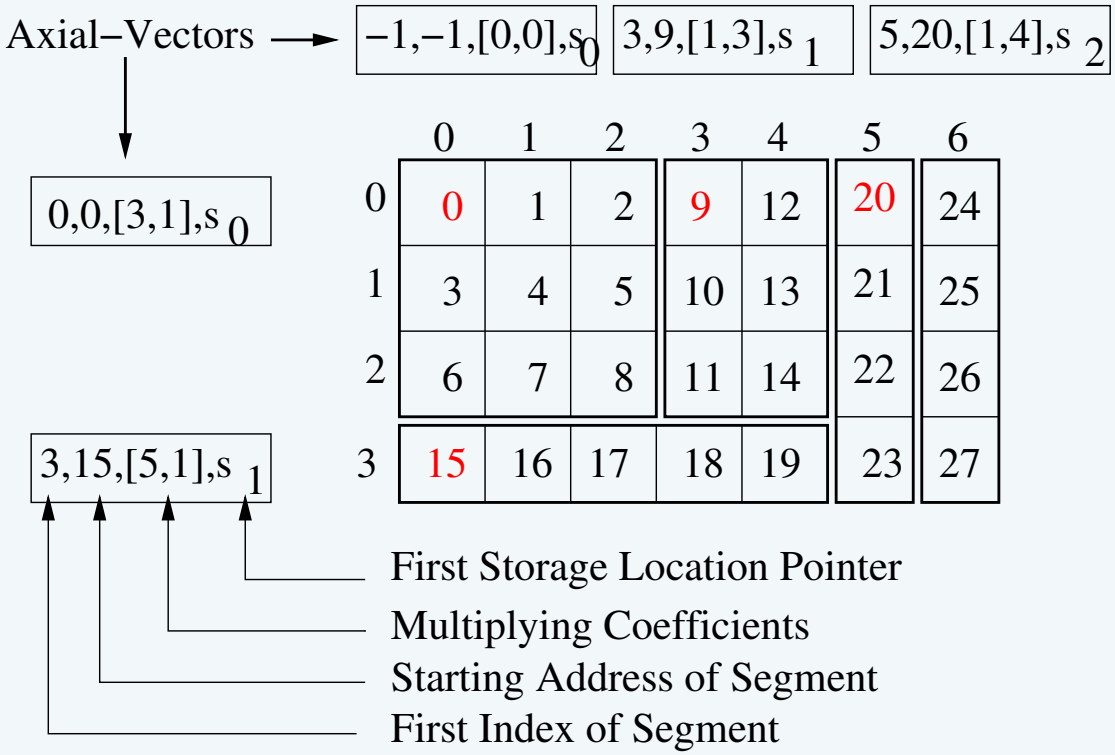
Illustration of an Extendible Array

- A 2-D array initially defined as $A[3][3]$ and then extended by by 2 columns, then by 1 row, followed by 1 column and so on.

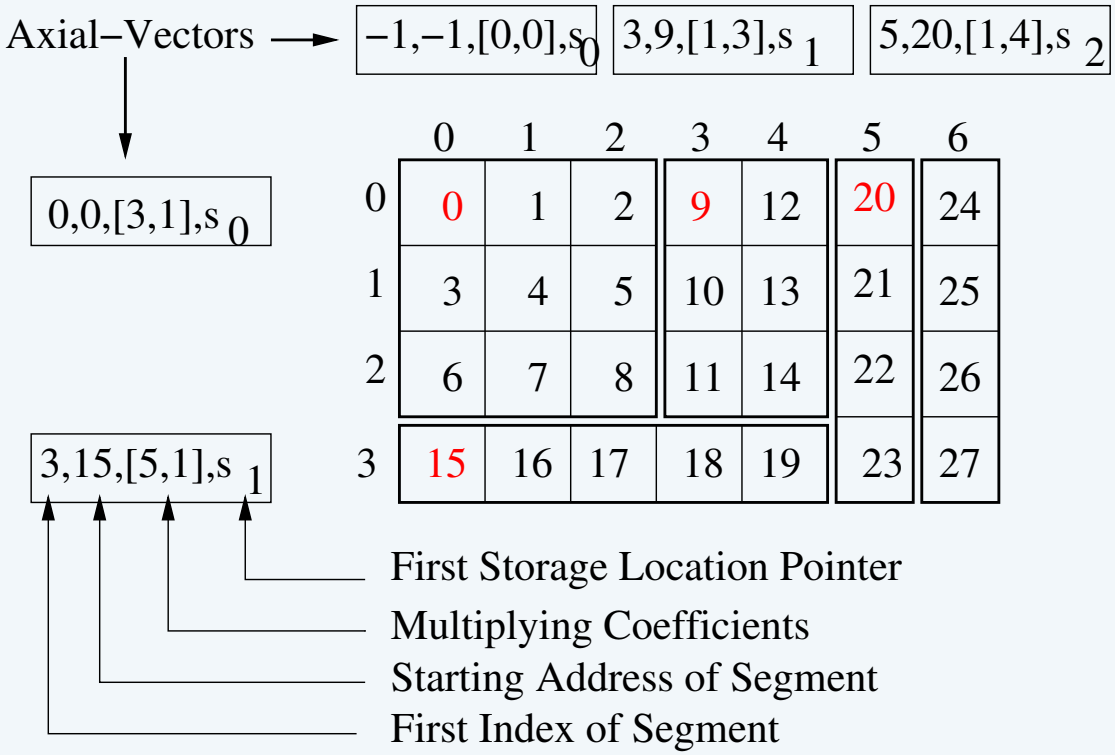
	0	1	2	3	4	5	6
0	0	1	2	9	12	20	24
1	3	4	5	10	13	21	25
2	6	7	8	11	14	22	26
3	15	16	17	18	19	23	27

- The labels in the cells are location addresses of the elements.
- An element $A\langle 2,5\rangle$ maps to location 22
- The address calculation is done by a function denoted as:
  $\mathscr{F}_*(i_0, i_1, \ldots, i_{k-1}) \rightarrow I$
  and an inverse $\mathscr{F}_*^{-1}(I) \rightarrow \langle i_0, i_1, \ldots, i_{k-1}\rangle$

Linear Mapping for an Extendible Array

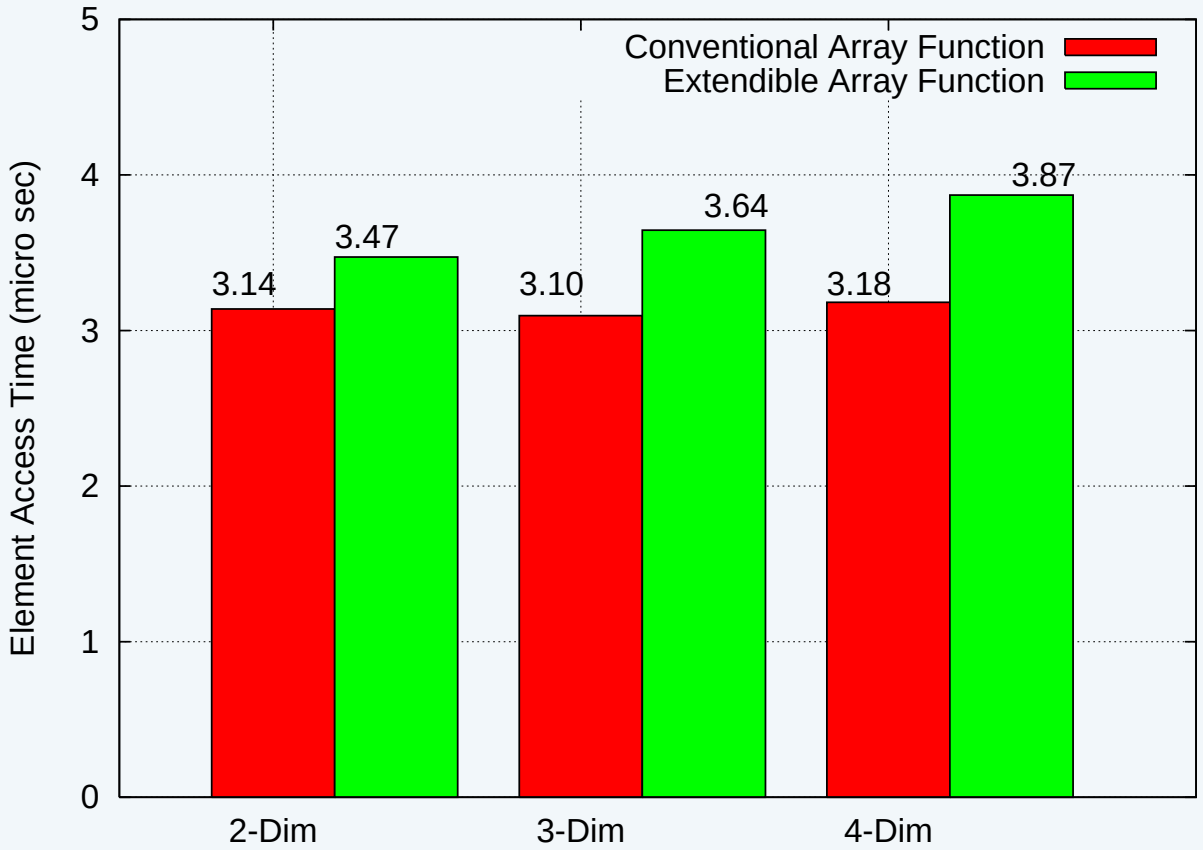


Linear Mapping for an Extendible Array



- The element $A\langle 2,5\rangle$ is located in either segment of row 2 with start address 0 or segment of column 5 with start address 20.
- It is always allocated in segment with maximum starting address.
- The address of $A\langle 2,5\rangle$ is computed by the algorithm $\mathscr{F}_*()$.

Comparison of File Element Access Cost



Disk Resident Extendible Arrays

- The elements are first grouped into *chunks* of some predefined *Chunk-Shape*, $A[I_0][I_1]\ldots A[I_{k-1}]$

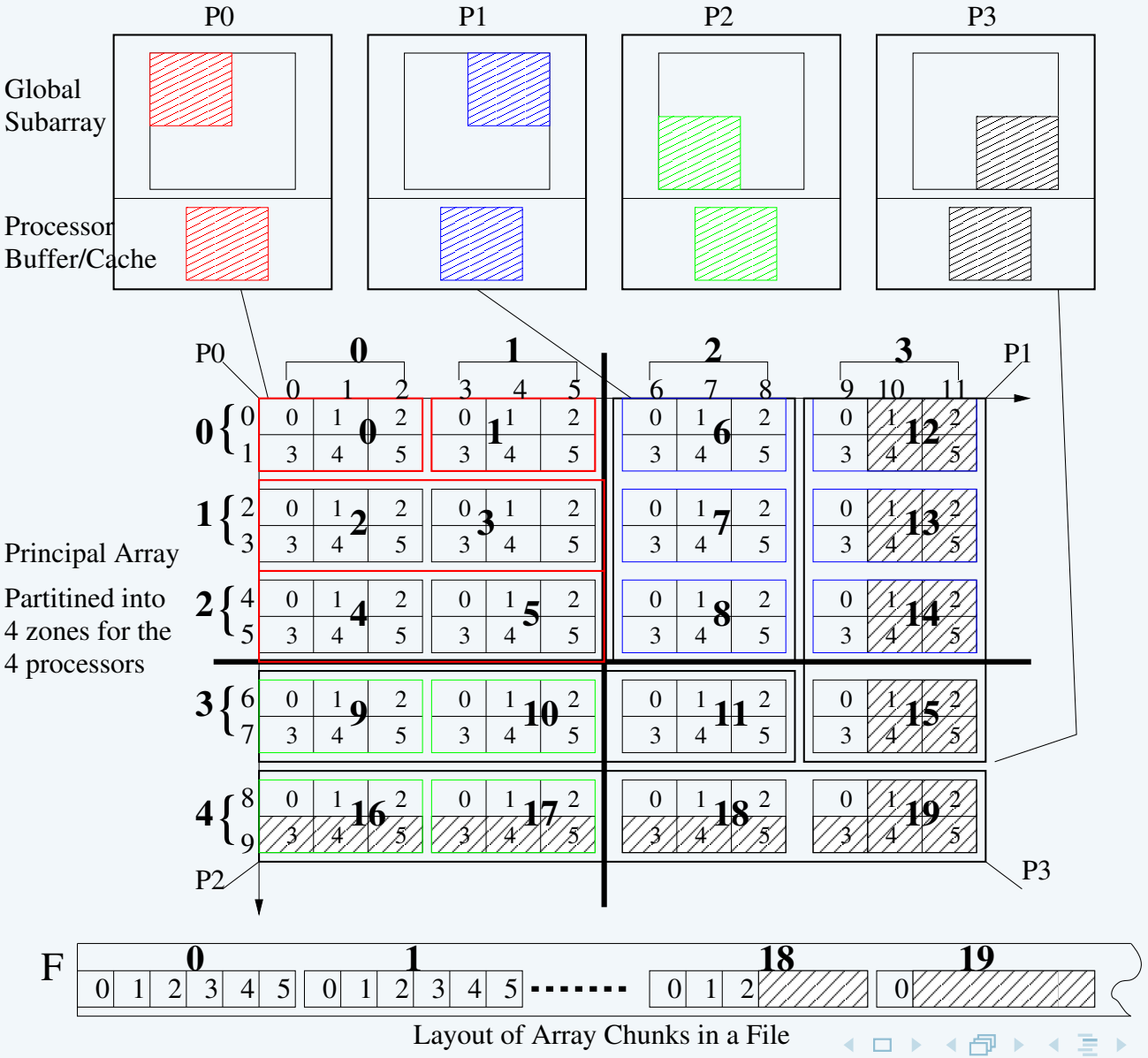
Disk Resident Extendible Arrays

- The elements are first grouped into *chunks* of some predefined *Chunk-Shape*, $A[I_0][I_1]\ldots A[I_{k-1}]$
- The chunks form the units of transfer between memory and a parallel file system.
- The mapping functions discussed are now applied to address the chunks and the array elements within a chunk can now be accessed using conventional array element address calculation.

Disk Resident Extendible Arrays

- The elements are first grouped into *chunks* of some predefined *Chunk-Shape*, $A[I_0][I_1]\ldots A[I_{k-1}]$
- The chunks form the units of transfer between memory and a parallel file system.
- The mapping functions discussed are now applied to address the chunks and the array elements within a chunk can now be accessed using conventional array element address calculation.
- The *Axial-Vectors* are retained in a Meta-Data file but read into memory at each session.
- Additional information in the Meta-Data include the bounds of the array, the chunk-shapes, etc.

The Allocation Scheme



Accessing Extendible Arrays (The pDRXA Library)

- Array is distributed by Block, Block partitioning scheme and along chunk boundaries. Block-Cyclic partitioning not yet
- A process controls a region of sub-array called a *zone* and an application can sub-arrays with either independent or collective I/O.

Accessing Extendible Arrays (The pDRXA Library)

- Array is distributed by Block, Block partitioning scheme and along chunk boundaries. Block-Cyclic partitioning not yet
- A process controls a region of sub-array called a *zone* and an application can sub-arrays with either independent or collective I/O.
- Each process then makes its zone accessible by creating a memory window for RMA access.
- Since each process has all the distribution information, it can access an element locally, if it controls the zone of the element; otherwise it accesses the element remotely via functions like *MPI_Get()*, *MPI_Put()* and *MPI_Accumulate(), etc.*

Accessing Extendible Arrays (The pDRXA Library)

- Array is distributed by Block, Block partitioning scheme and along chunk boundaries. Block-Cyclic partitioning not yet
- A process controls a region of sub-array called a *zone* and an application can sub-arrays with either independent or collective I/O.
- Each process then makes its zone accessible by creating a memory window for RMA access.
- Since each process has all the distribution information, it can access an element locally, if it controls the zone of the element; otherwise it accesses the element remotely via functions like *MPI_Get()*, *MPI_Put()* and *MPI_Accumulate(), etc.*
- The processing model is consistent with the Global-Array toolkit model for parallel processing of arrays.
- The idea then is to define the access functions to be consistent with the Disk Resident Array library of GA and leverage the scientific processing capability of GA.

Future Work Plans

- There are a number of popular and established standard array oriented file formats that the methods can be used to support.
  - HDF4/HDF5 and their derivatives - HDF5-EOS, HDF5-Lite.
  - HDF5 allows extendibility in any dimension using data chunking and manages the chunks using a B-Tree index.
  - The B-Tree index can be replaced with DRXA access schemes

Future Work Plans

- There are a number of popular and established standard array oriented file formats that the methods can be used to support.
  - HDF4/HDF5 and their derivatives - HDF5-EOS, HDF5-Lite.
  - HDF5 allows extendibility in any dimension using data chunking and manages the chunks using a B-Tree index.
  - The B-Tree index can be replaced with DRXA access schemes
- DRA is the persistent storage for GA and we mimic its access methods in pDRXA.
- An API of pDRXA consistent with those of DRA should make it accessible to any application that uses GA.

Future Work Plans

- There are a number of popular and established standard array oriented file formats that the methods can be used to support.
  - HDF4/HDF5 and their derivatives - HDF5-EOS, HDF5-Lite.
  - HDF5 allows extendibility in any dimension using data chunking and manages the chunks using a B-Tree index.
  - The B-Tree index can be replaced with DRXA access schemes
- DRA is the persistent storage for GA and we mimic its access methods in pDRXA.
- An API of pDRXA consistent with those of DRA should make it accessible to any application that uses GA.
- NetCDF {NCAR} and parallel NetCDF {SDM Center} can utilize pDRXA access functions without reliance on HDF5