BERKELEY LAB
Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY
Office of Science

FasTensor and Its Applications in Geoscience

Bin Dong, PhD, Research Scientist, Lawrence Berkeley National Lab

Career @ LBNL

Research Scientist , Scientific Data Division, LBNL, **2016 - Present**
Postdoctoral Research Fellow, Scientific Data Division , LBNL, **2013 - 2016**

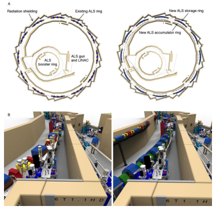
Research Interests
Bin's research interests are in Big scientific data analysis, parallel computing, machine learning. Bin is exploring new and scalable algorithms and data structures for sorting, organizing, indexing, searching, analyzing Big Array Data with supercomputers.

This talk about how to analyze big array data from scientific applications, specifically

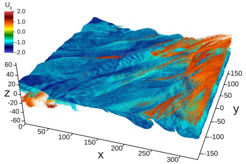
- Part 1: How to build a generic array data analysis system — FasTensor
- Part 2: How to apply it to a geoscience application — DASSA

Part 1: FasTensor

Mountains of Scientific Data Wait for Analysis

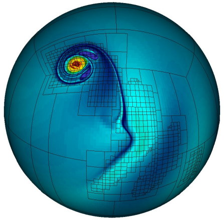


Light Source
180 PB/year
(ALS-U at Berkeley Lab)

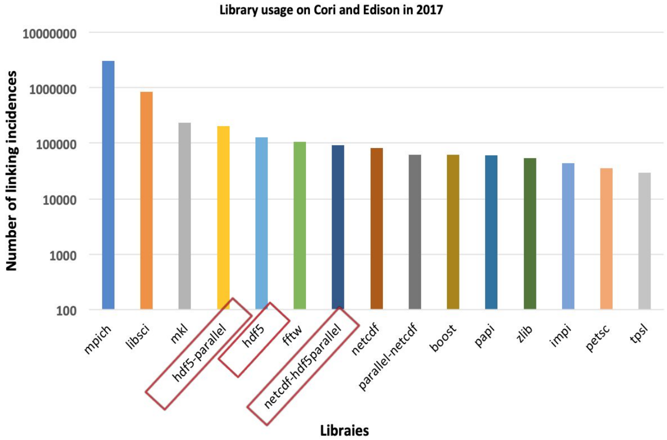


High Energy Physics
200 PB/year

Genomics
10 PB/year



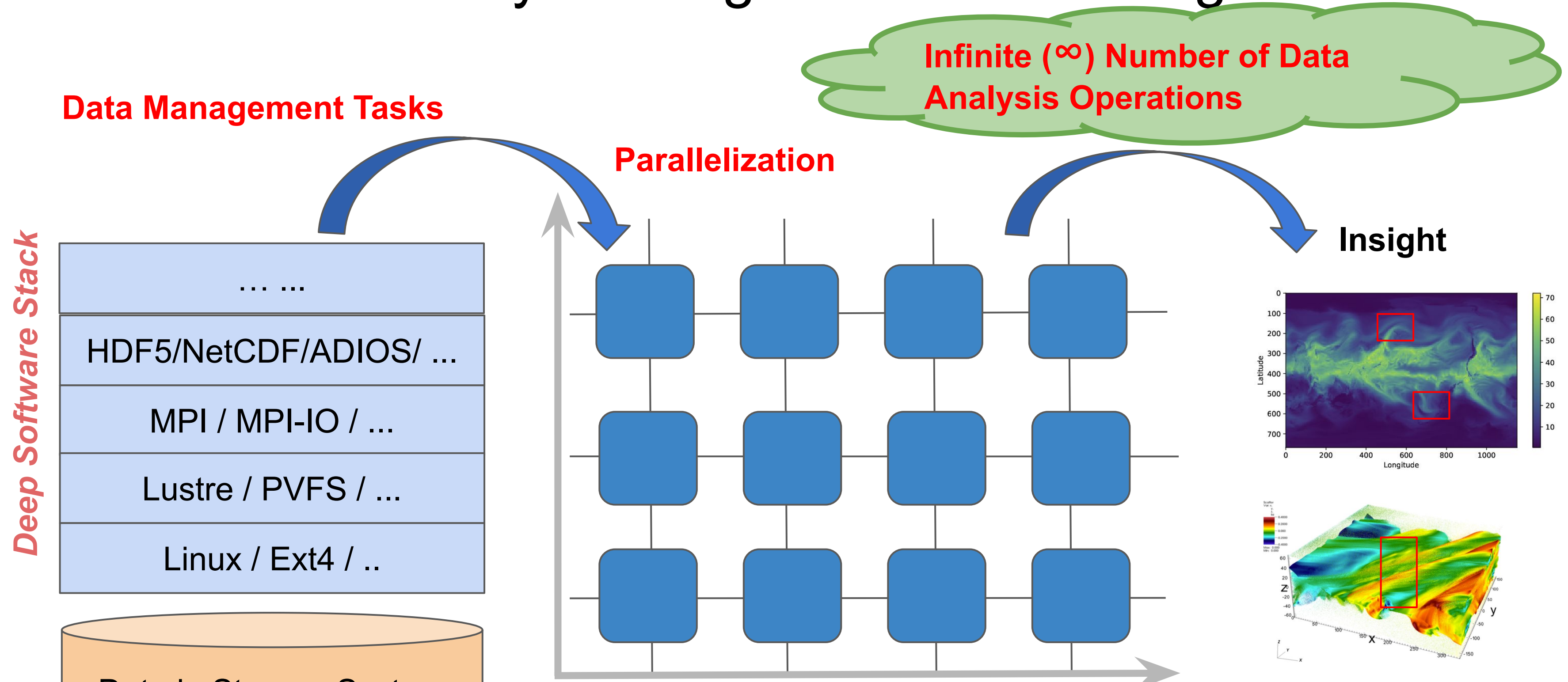
Climate
100 EB/year



Most are multidimensional arrays, stored in file formats like HDF5, PNetCDF, ADIOS, etc

Sources: L. Nowell, D., Ushizima, S. Byna, JGI and ALS at LBNL, etc.

Factors to Consider During the Development of Scientific Data Analysis Programs to Find Insight



Many Computing Nodes and CPU Cores (over 7 million cores at FUGAKU, No. 1 of Top500 on Nov 2021)

Two Common Methods to Develop Scientific Data Analysis Programs

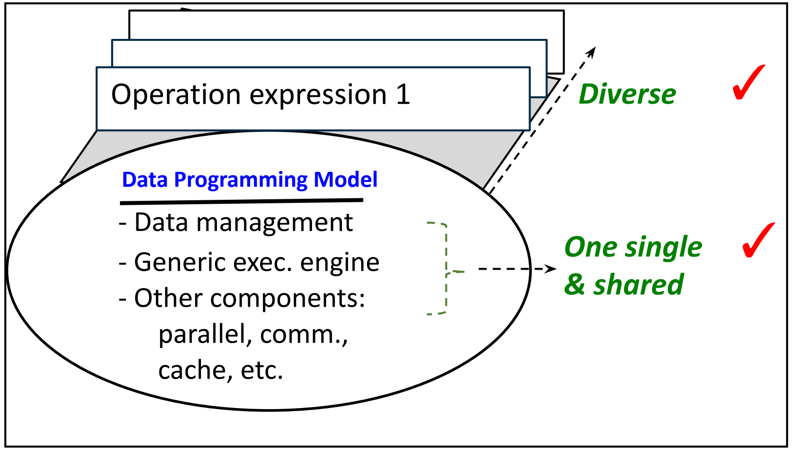
Customized Solutions

For each operation *P* **Do**
Develop *P*'s :
- Data management → *Redundant* ✗
- Expression execution → *Diverse* ✓
- Other components: parallel, communication cache, etc. → *Redundant* ✗

End For

May lack expertise of the underlying systems to tune its performance ✗

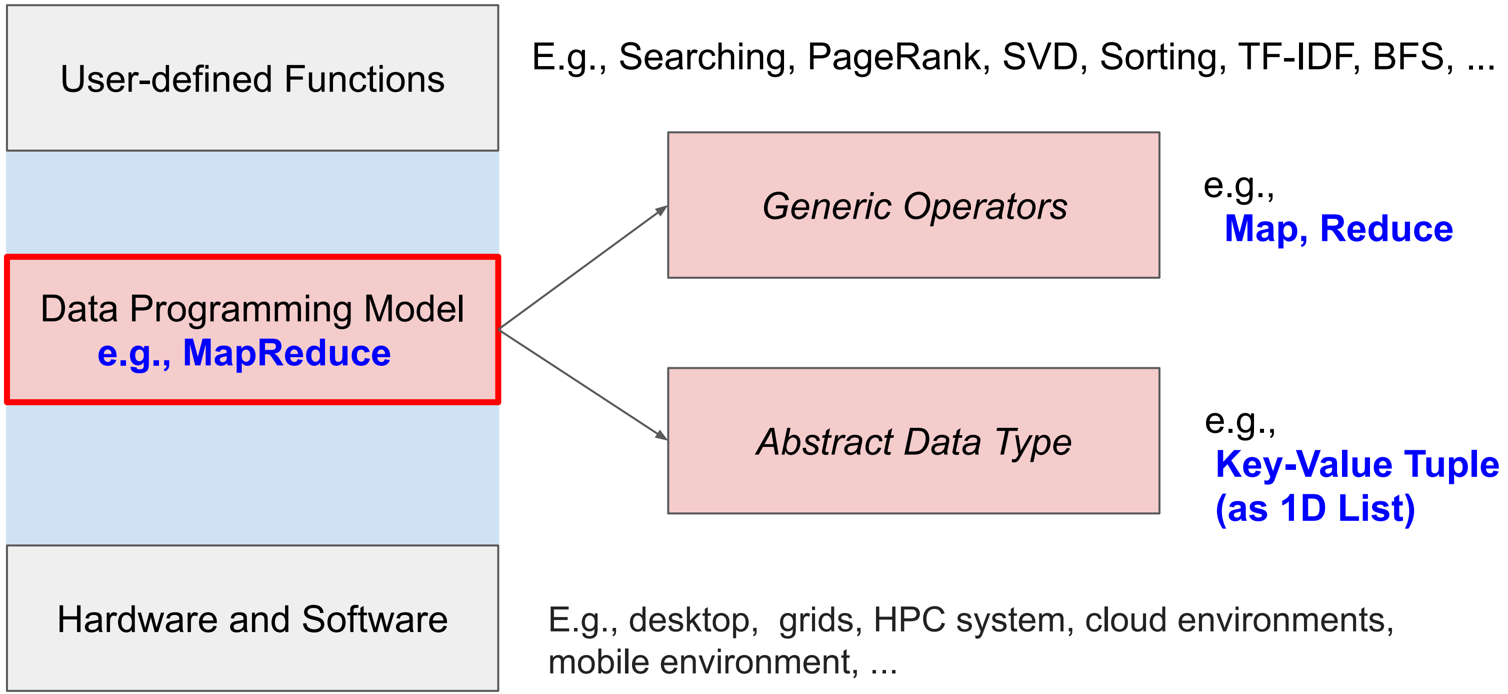
User-defined Functions (UDF)



Professionally tuned ✓

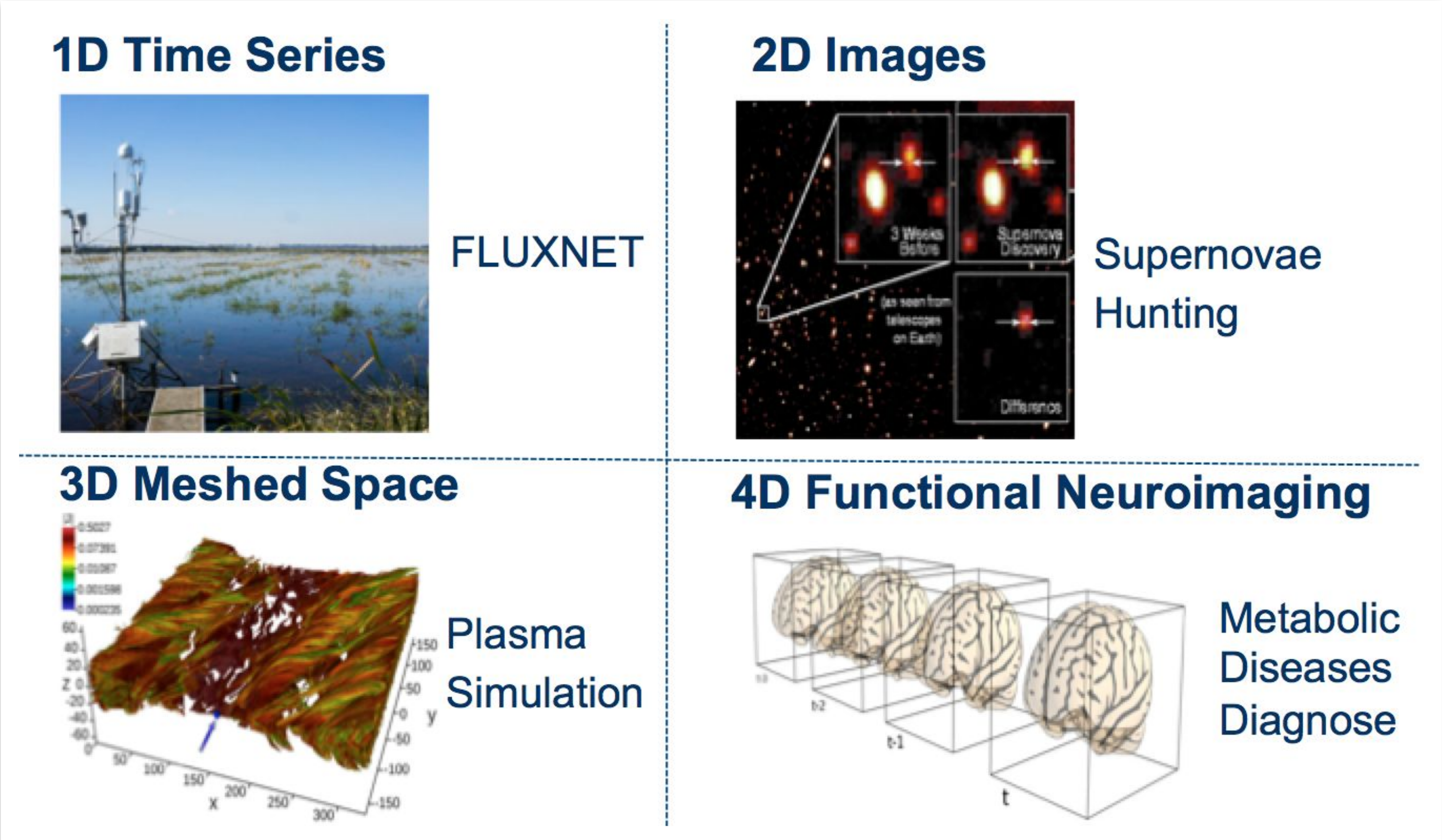
MapReduce Data Programming Model

Data programming model: an data programming abstraction, hiding complexities of hardware/software and being generic to many advanced analysis tasks



Tensor is Dominated Data Structure in Science

- Tensor is defined as multidimensional array here



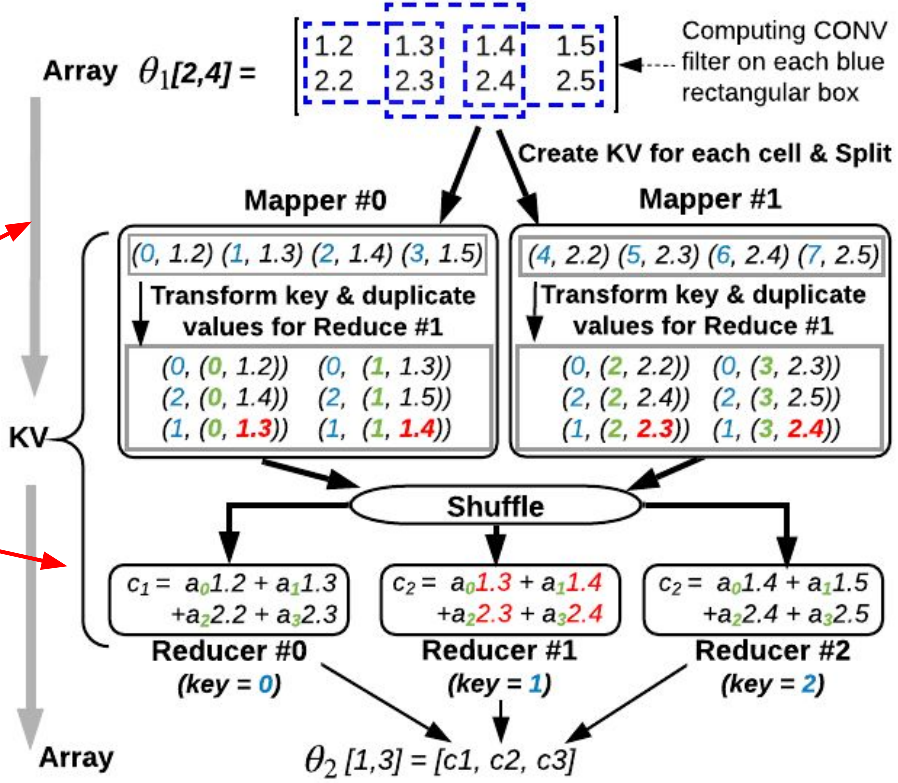
MapReduce's Limitations in Tensor Data Analysis

Convolution on a 2 by 4 2D Tensor (Array)

Kernel is 2 by 2

1, Mismatched Data Model
-- Convert Tensor to KV list at Map stage

2, Expensive reduce
-- Duplicate KV for Reduce stage



FasTensor: new data programing model on array

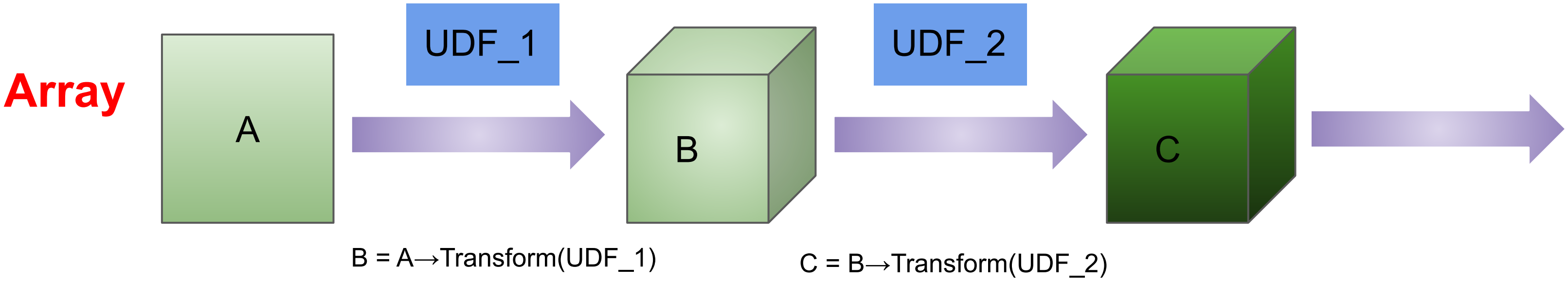
Inspiring by: a Tensor can be defined as a multidimensional array and proper transformation rules

FasTensor is a generic parallel data programming model

Tensor = Multidimensional **Array** + **Transform** Rules

Data Model

User Defined Function (UDF)



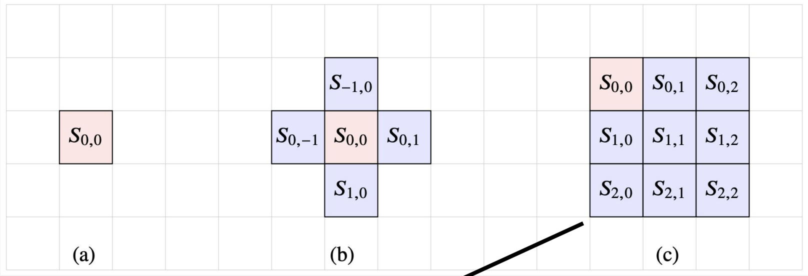
How FasTensor works

- ➔ Multidimensional **Array** Model
  - Disk (e.g. HDF5, ADIOS, netCDF)
  - Memory (e.g., DASH)
- ➔ Flexible **Stencil** Data Structure
  - Flexible UDF functions
- ➔ Execution Engine
  - Auto-parallel: MPI/OpenMP/hybrid
  - Optimized Chunking Size
  - Optimized Overlap
  - In-place Modification Semantic
  - Fault-tolerance Support
  - etc.

Stencil

Base Cell

Neighbor Cells -- relative offset(s)



Example: define a sequential **Sum** as udf_Window_Aggregates()

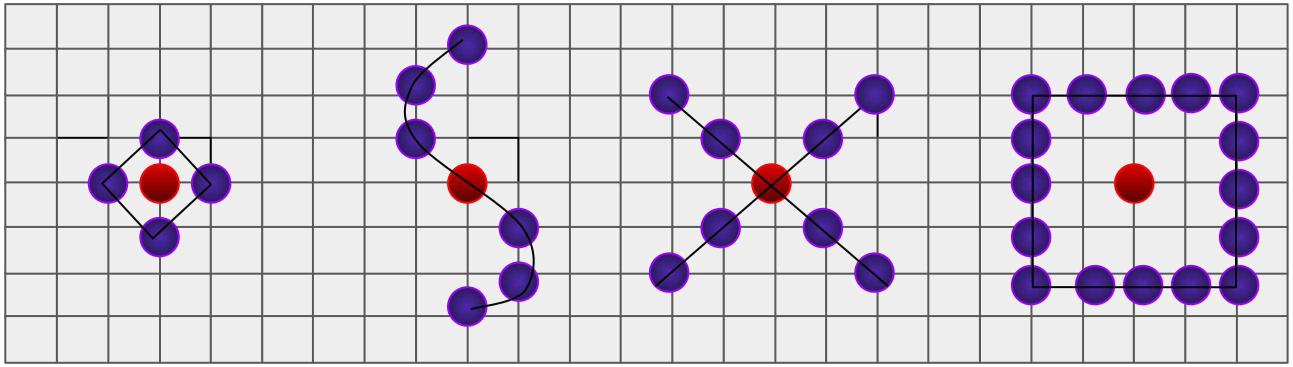
```
inline Stencil<float> udf_Window_Aggregates(const Stencil<float> &iStencil)
{
    Stencil<float> oStencil;
    oStencil =   iStencil(0, 0) + iStencil(0, 1) + iStencil(0, 2)
               + iStencil(1, 0) + iStencil(1, 1) + iStencil(1, 2)
               + iStencil(2, 0) + iStencil(2, 1) + iStencil(2, 2);
    return oStencil;
}

int main(int argc, char *argv[]){
   B = A->Transform(udf_Window_Aggregates)
}
```

Stencil: Abstract Data Type

Stencil

- An abstract data structure to represent a neighborhood of an Array
- Definition: ***S***(Base Cell, Neighbor Cells -- relative offsets)



Structural Locality

Flexible geometric shapes/size to break an array
into small units for atomic, out-of-core, or parallel processing

FasTensor Programming Model

Data Model: Multi-dimensional Array

Abstract Data Type: Stencil

Generic Operator: Transform

v.s. MapReduce

- Data Model: 1D List
- Abstract Data Type: Key-Value Pair
- Generic Operator: Map, Reduce

User-Defined Function (UDF): Rules of the Transform

Transform: $S_1 \overset{f}{\mapsto} S_2 \, , \quad S_1 \subset A \, , \quad S_2 \subset B$

Stencil (S_1, S_2) Array (A, B)

An Example of 3-point Moving Average

$$V = \frac{V_{t-1} + V_t + V_{t+1}}{3}$$

```
int main(int argc, char *argv[])
{
    //Init the MPICH, etc.
    FT_Init(argc, argv);
    // set up the chunk size and the overlap size
    std::vector<int> chunk_size = {4, 4};
    std::vector<int> overlap_size = {0, 1};
    //Input data
    Array<float> A("EP_HDF5:tutorial.h5:/data", chunk_size, overlap_size);
    //Result data
    Array<float> B("EP_HDF5:tutorial_ma.h5:/data");
    //Run
    A.Transform(udf_ma, B);
    FT_Finalize();
    return 0;
}
inline Stencil<float> udf_ma(const Stencil<float> &iStencil)
{
    Stencil<float> oStencil;
    oStencil = (iStencil(0, -1) + iStencil(0, 0) + iStencil(0, 1)) / 3.0;
    return oStencil;
}
```

Input Array A, a 2D 16 x 16 dataset in HDF5 file, where each row is a time series from a sensor

Output Array B, a 2D dataset in HDF5 file

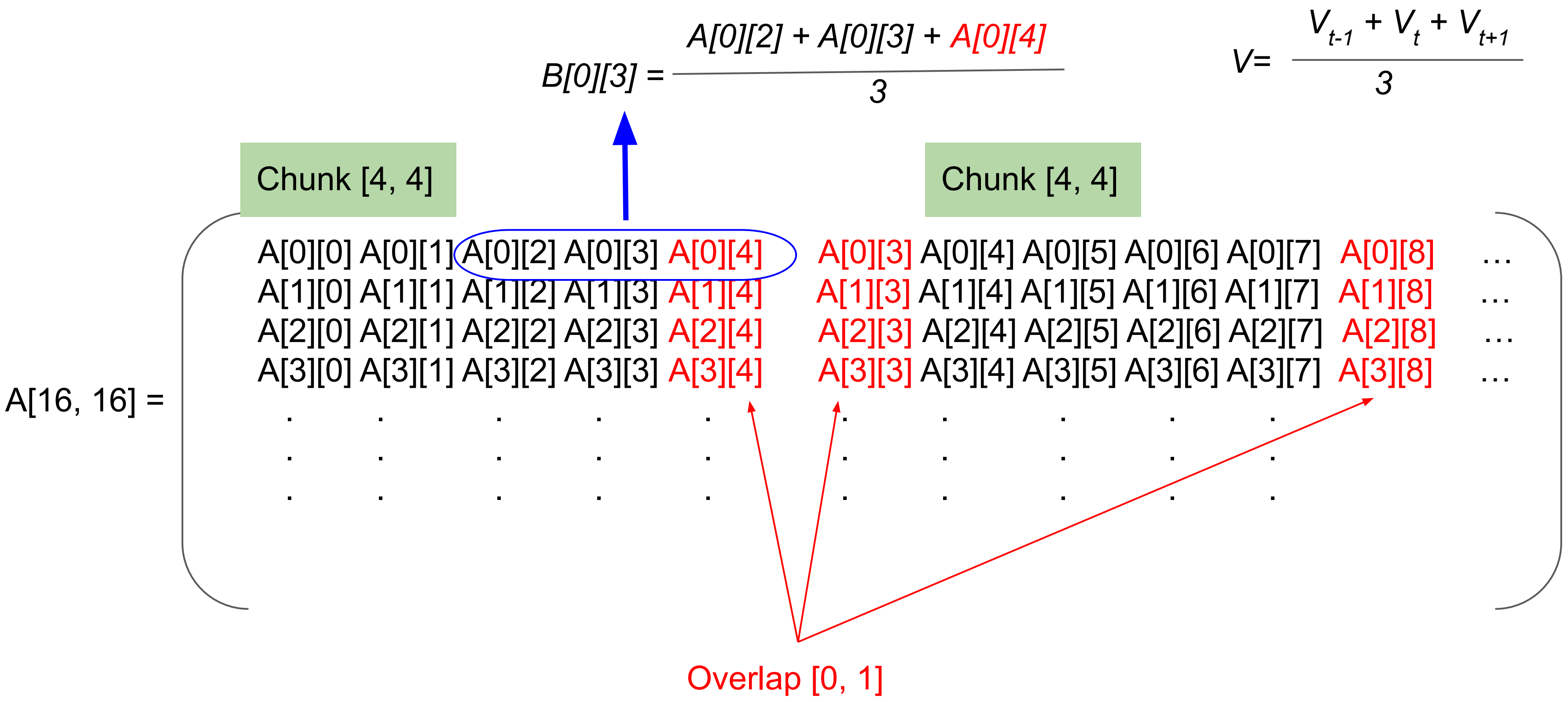
Execute the Transform, either sequentially or in parallel

Rules of the Transform from A to B

$$\frac{V_{t-1} + V_t + V_{t+1}}{3}$$

Relative offsets

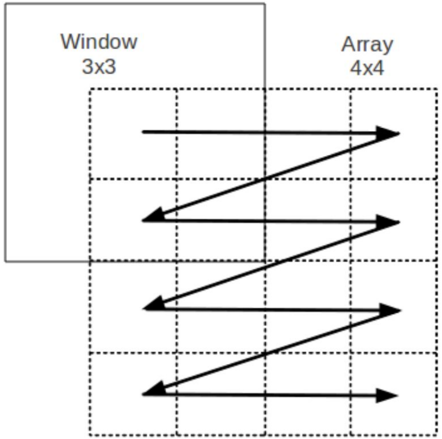
Overlap in FasTensor to Avoid Communication During Execution of UDF



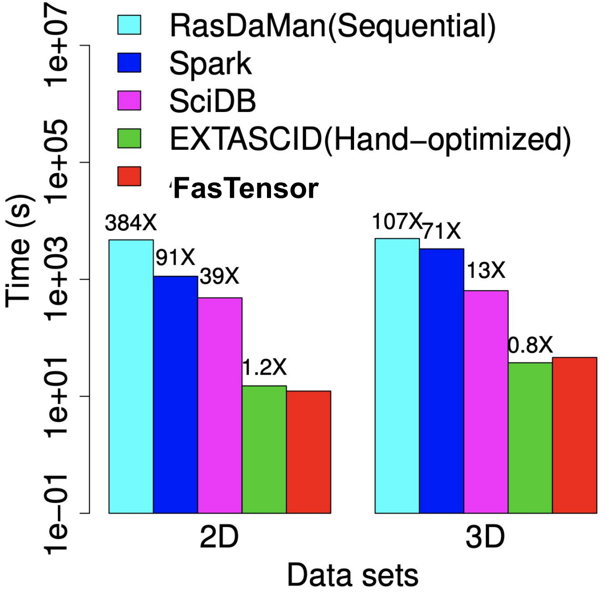
DBMS Data Analysis

v.s. SciDB, Spark and RasDaMan

Window Aggregates *(scidb-userguide, sec 7.4)*



Problems in SciDB:
- Fixed Rectangular Window
- Duplicate Data across Windows
  (also in Spark)



(Bin Dong, Kesheng Wu, Surendra Byna, etc. HPDC '17.)

In AQL, you would use this statement:

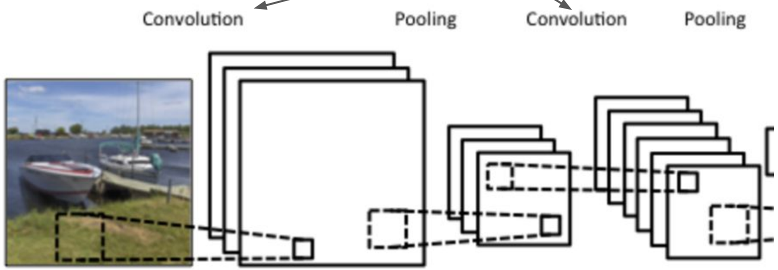
```
AQL% SELECT sum(attr1)
     FROM m4x4
     WINDOW 3,3;
```

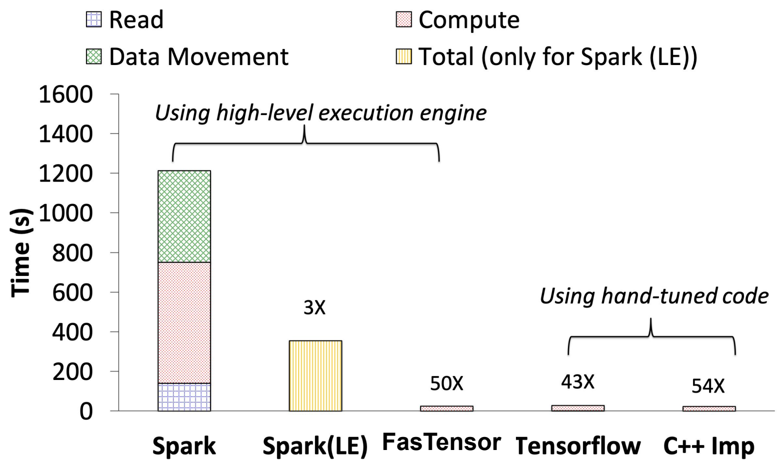
Machine Learning (AI)

v.s. TensorFlow, Spark

Convolution in Convolutional Neural Network (CNN) (forward step)

Expensive: convolution takes 65% of the forward step

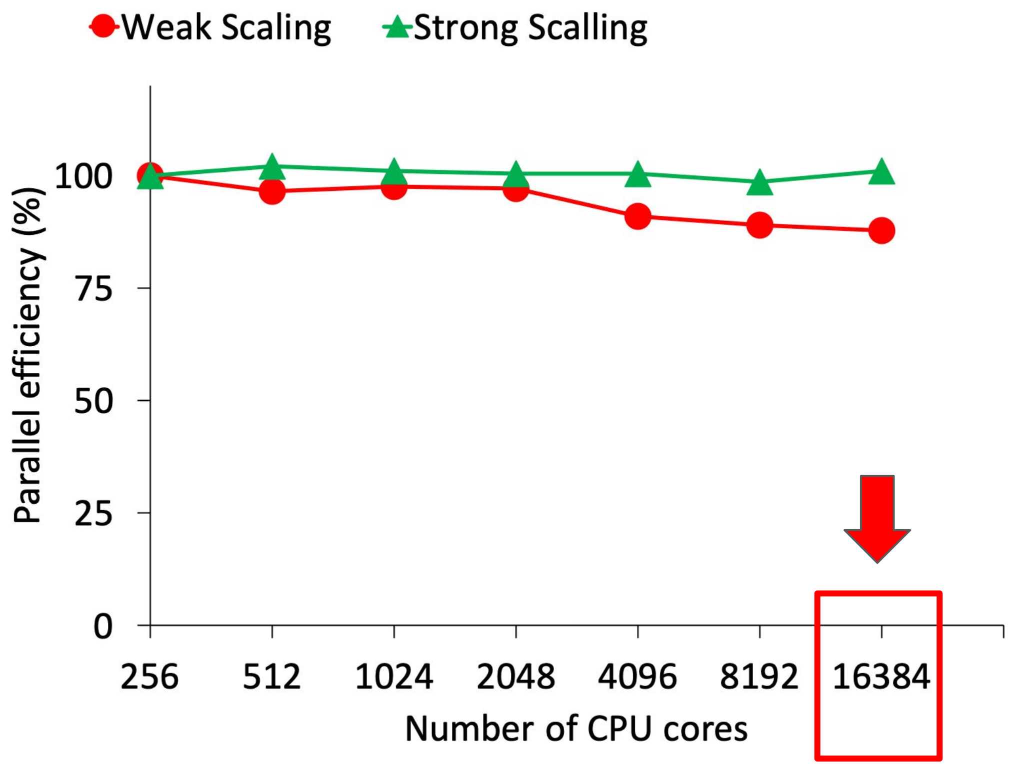




(Bin Dong, etc, ISC 2019)

Problems in Existing System:
- TensorFlow may use expensive matrix multiplication
- Spark may duplicate data across convolution Filters

FasTensor Scales Perfectly on CNN Computation

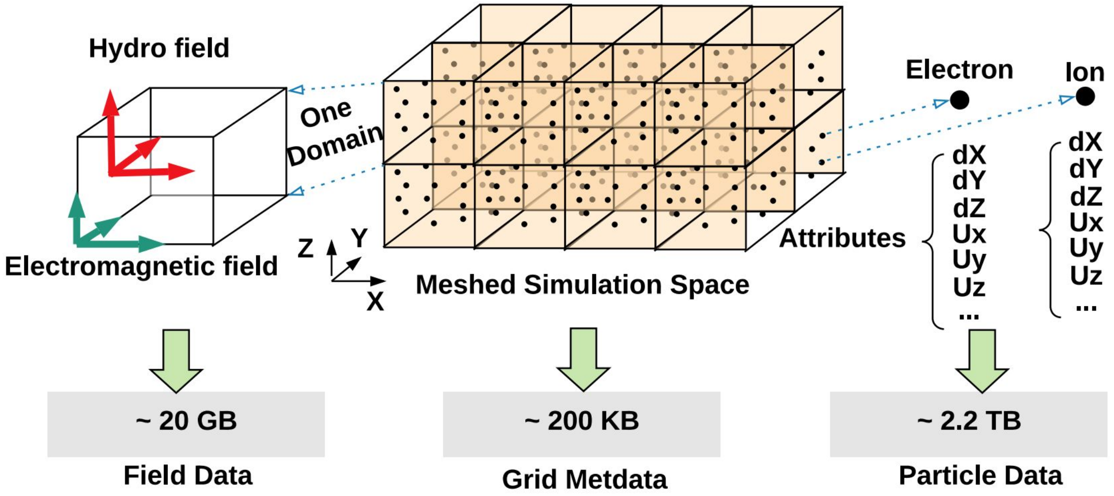


Parallel efficiency =

$$\begin{cases} t_1/t_N * 100\% & \text{Weak Scaling} \\ t_1/(N * t_N) * 100\% & \text{Strong Scaling} \end{cases}$$

t_i is the time to finish the work with i CPU cores

FasTensor in Plasma physics (VPIC)

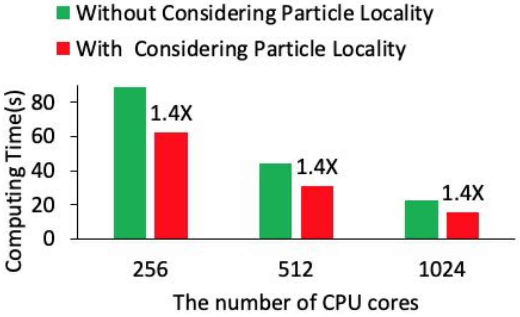


Bin Dong, Patrick Frank Heiner Kilian, Xiaocan Li, Fan Guo, Suren Byna and Kesheng Wu, "Terabyte-scale Particle Data Analysis: An ArrayUDF Case Study", SSDBM 2019, July 23, 2019,

$$
\begin{aligned}
E_{px} =& (1-dY)(1-dZ)E_x(i\times d_x,(j-0.5)\times d_y,(k-0.5)\times d_z)/4+\\
& (1+dY)(1-dZ)E_x(i\times d_x,(j+0.5)\times d_y,(k-0.5)\times d_z)/4+\\
& (1-dY)(1+dZ)E_x(i\times d_x,(j-0.5)\times d_y,(k+0.5)\times d_z)/4+\\
& (1+dY)(1+dZ)E_x(i\times d_x,(j+0.5)\times d_y,(k+0.5)\times d_z)/4
\end{aligned}
$$

- *Field Data Analysis*
- *Particle Data Analysis*
- *Fusing Particle Data and Field Data*

FastTensor could easily express all these operations



Time to Fuse Fields and Particles Data

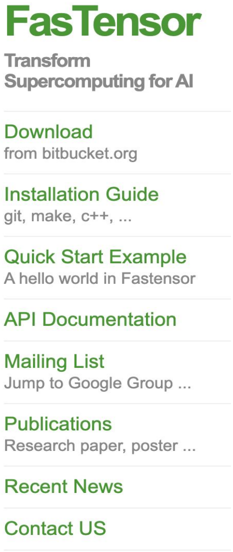
Key Takeaways

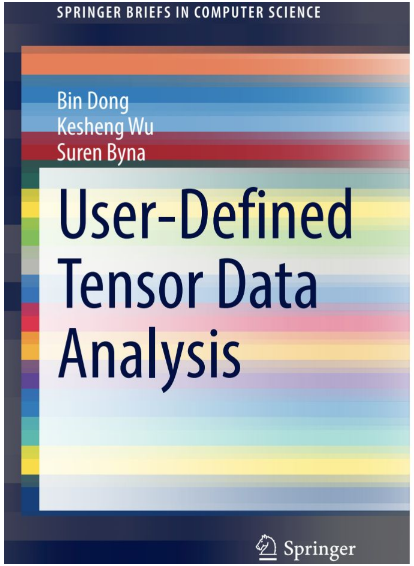
- ➔ FasTensor is a generic parallel data programming model for multidimensional array
- ➔ FasTensor supports auto parallelization /auto-chunking/etc on HPC system
- ➔ FasTensor can be 100X faster than Spark in executing array-based data analysis
- ➔ FasTensor can be used by different types of data analysis, e.g., plasma, DBMS, AI

Resources

Website: https://sdm.lbl.gov/fastensor/

Book: User-defined Tensor Data Analysis





Part 2: DASSA

Parallel DAS Data Storage and Analysis for Subsurface Event Detection

Bin Dong[1], Verónica Rodríguez Tribaldos[1], Xin Xing[2],
Suren Byna[1], Jonathan Ajo-Franklin[1,3], Kesheng Wu[1]

[1]Lawrence Berkeley National Laboratory, Berkeley, CA, USA

[2]Georgia Institute of Technology, Atlanta, GA, USA

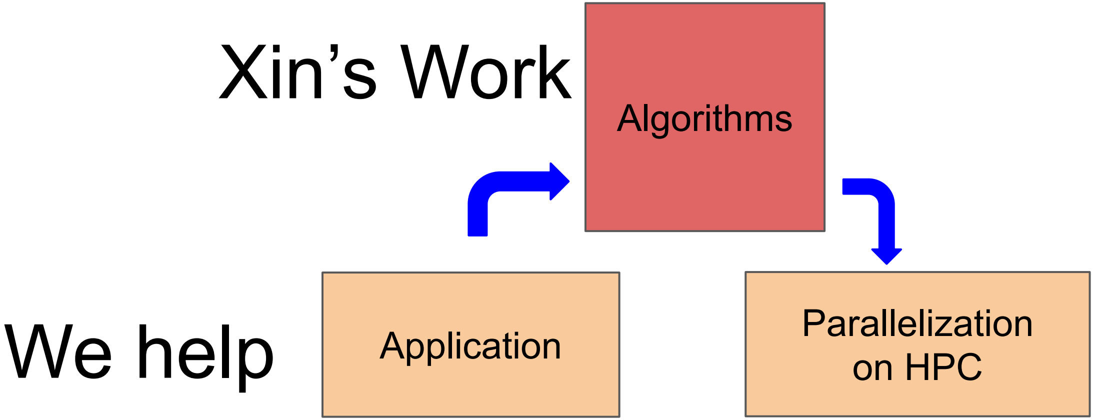
[3]Rice University, Houston, TX, USA

Xin Xing, summer student @ 2018

- A PhD student at School of Mathematics, Georgia Tech

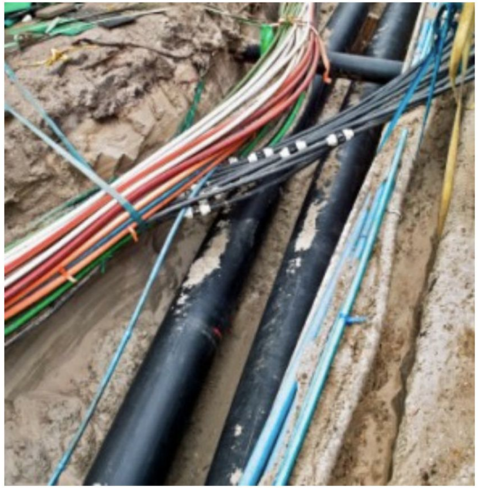
  (Now postdoc @ LBNL)

- Two papers together
  - ➔ Automated Parallel Data Processing Engine with Application to Large-Scale Feature Extraction", (MLHPC) in SC 2018
  - ➔ "DASSA: Parallel DAS Data Storage and Analysis for Subsurface Event Detection", IPDPS 2020



Dark Fiber: unused optical fibre, everywhere

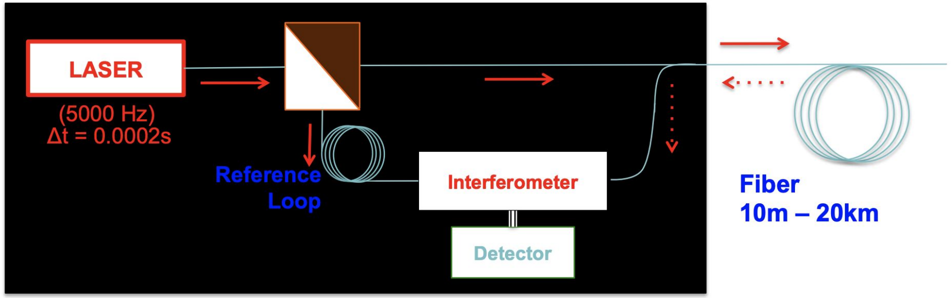
How can we use it for science?



DAS repurposes it as subsurface monitors

DAS: Distributed acoustic sensing

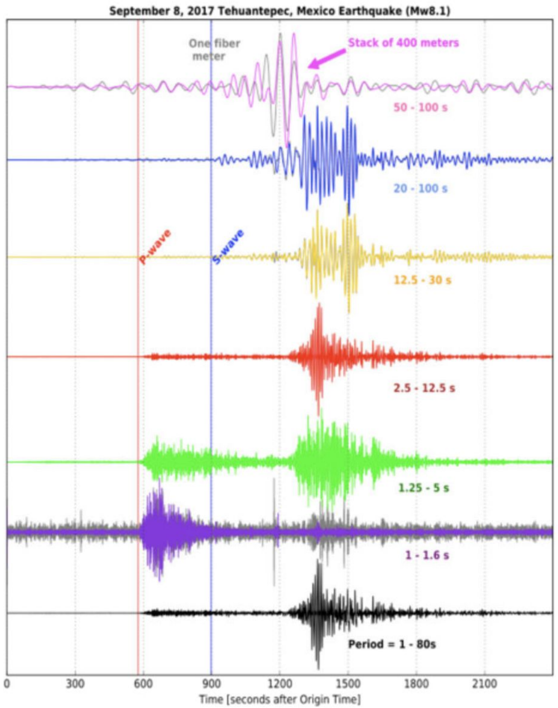
How does DAS work?



1. Shoot laser pulse into fiber-optic
2. Track phase in a reference loop
3. Rayleigh backscattering from “scatterers”
4. Phase-based interferometry vs. time
5. Time → distance L
6. Repeat at 5-10khz

DAS data are natively strain-rate: $\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}\right) = \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial t}\right)$

Hartog et al., (2014)
Daley et al., (2016)

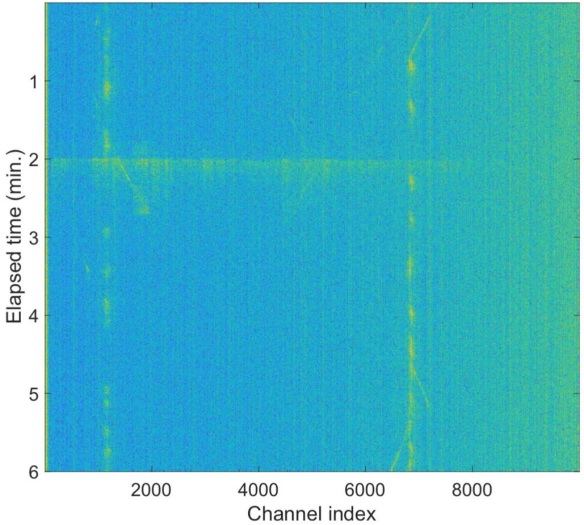


DAS and its Data Analysis Challenges

DAS: Distributed acoustic sensing

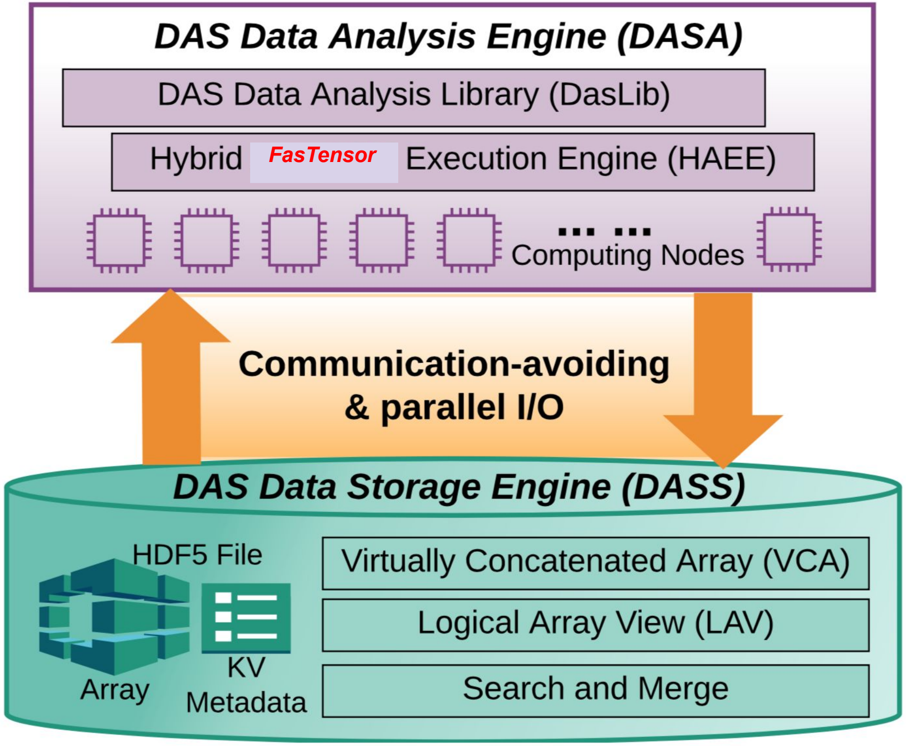
- record strain or strain-rate along fiber-optic cables in subsurface
- wide application in geophysical, e.g., earthquake detection, seismic imaging

Data Analysis Challenges



- DAS data size is large (TB/data), but scattered among many small but dense arrays
- Different analysis operations are required in different DAS data investigations.

Our solution: DASSA framework



ASSA: a scalable and easy-to-use system for eophysicists to perform DAS data analysis on HPC

Data Data Analysis Engine:

- DasLib: sequential DAS data analysis operations
- HAEE: a transparently parallelization engine (FasTensor) for DasLib's operations

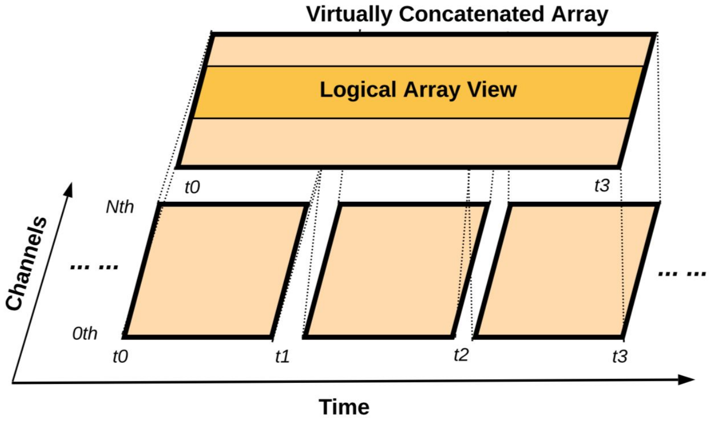
Data Data Storage Engine

- Array and KV based Data Model
- VCA for analysis across many small files
- Other support functions, logical view and search and merge

I/O optimization methods through avoiding communication during I/O stage of analysis

DASS: DAS data storage engine

Virtual Concatenated Array Data Model



Concatenate small files for analysis

- Easy to be parallelized for execution engine
- Without duplicates during construction
  - Metadata based construction

Key-value based Metadata Model

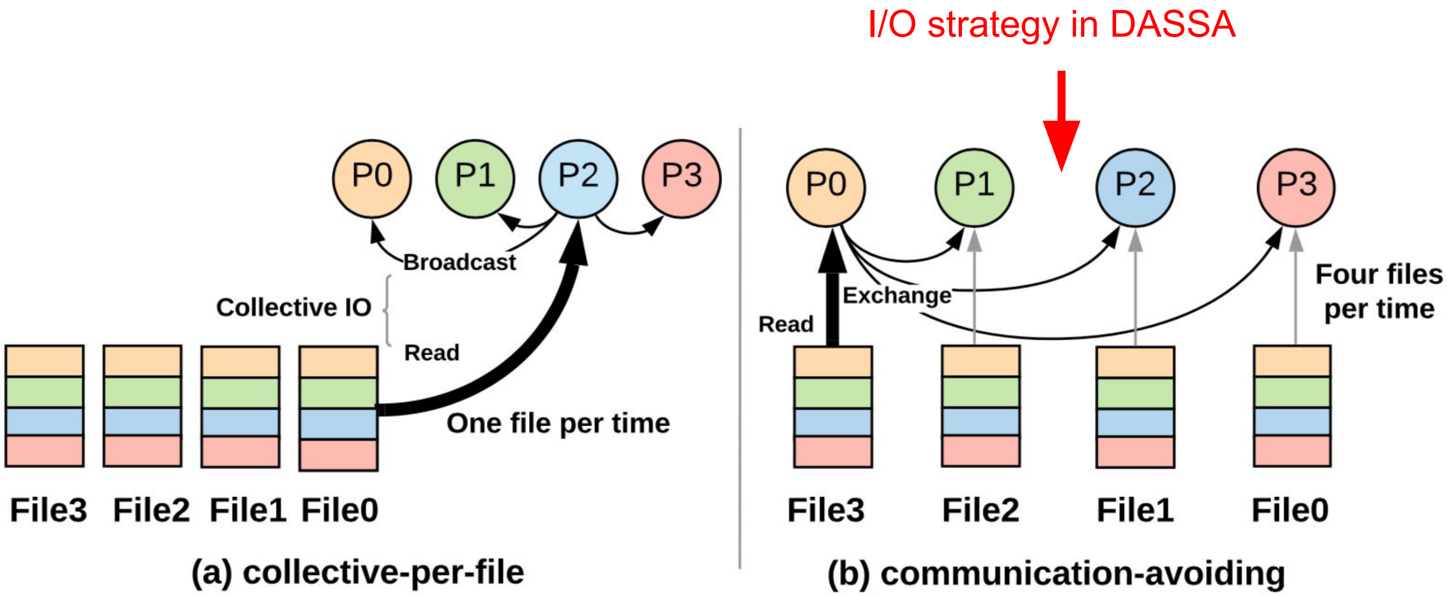
Root of DAS metadata in HDF5 file

- SamplingFrequency(HZ) : 500
- SpatialResolution(m) : 2
- TimeStamp(yymmddhhmmss) : 17062
- Number of objects : 11648
- ... *other global metadata* ...
- Object Path: /Measurement/1
  - Array dimension : 1
  - Number of raw data values: 45
  - ... *other metadata per object* ...
- ... *other objects and their metedata* ...

Two levels of KV list

- Flexible in preserving metadata

I/O optimization in DASSA to reduce cost



a) “collective-per-file” method: all processes read a file at a time with collective-I/O per file.
b) “communication-avoiding” method: each process reads a file and then, all processes have an all-to-all data exchange to obtain their own data portion.

DASA: DAS data analysis engine

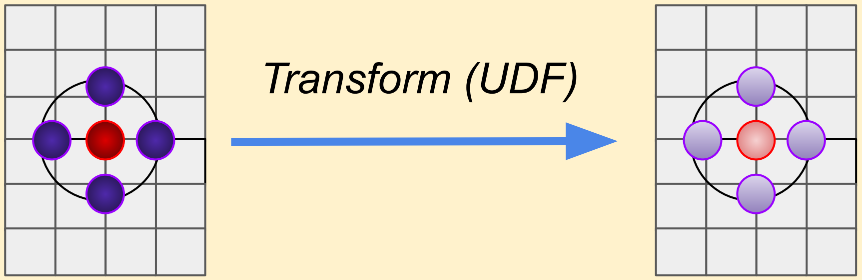
- **DasLib: sequential analysis operation**

Functions	Semantic
Das_abscorr(c_1, c_2)	absolute correlation of c_1 and $c2$ defined as $\lvert \cos(\theta(c_1, c_2)) \rvert$
Y = Das_detrend(X)	removes the best straight-line fit of x
(c_1, c_2) = Das_butter(n, f_c)	create Butterworth filter coefficients c_1 and c_2 with the cutoff frequency f_c
Y = Das_filtfilt(c_1, c_2, X)	apply c_1 and c_2 to X
Y = Das_resample(X, 1, R)	samples the X with new rate R
Y = Das_interp1(X_0, Y_0, X)	linearly interpolates f that satisfies $f(X_0) = Y_0$ to obtain the values Y at X
Y = Das_fft(X)	perform FFT on X
Y = Das_ifft(X)	perform inverse FFT on X

- **Parallel Execution Engine**
  - A Multithreaded extension of FasTensor

FasTensor

- An *Array*-native Data Programming Model
- A *Stencil*-based Abstract Data Type
- A single *Transform* operation
- HPC friendly



Data Analysis Examples

Earthquake detection [1] via local similarity

Algorithm 2 Local similarity calculation within HAEE as the user-defined function on DAS data.

Note: S is the Stencil abstraction representing an abstract cell and its neighborhood. Each window (e.g., W, W_1, and W_2) has width $(2M+1)$. Two neighboring channels have offsets $+K$ and $-K$ relative to the central channel, respectively. $(2L+1)$ windows are sampled on each neighboring channel (i.e., W_1 and W_2).

function LOCALSIMI(S)
  $W = S(-M:M, 0)$ ▷ Extract current window via S
  $C_{+K} = C_{-K} = 0$ ▷ initialization
  for $l = -L : L$ **do**
    $W_1 = S((l-M):(l+M), +K)$
    $W_2 = S((l-M):(l+M), -K)$
    $C_{+K} = \max\{C_{+K}, \text{Das\_abscorr}(W, W_1)\}$
    $C_{-K} = \max\{C_{-K}, \text{Das\_abscorr}(W, W_2)\}$
  end for
  return $\frac{1}{2}(C_{+K} + C_{-K})$ ▷ Local similarity.
end function

[1]Z. Li, Z. Peng, D. Hollis, L. Zhu, and J. McClellan. High-resolution seismic event detection using local similarity for large-n arrays. Scientific reports, 8(1):1646, 2018.

Traffic-noise interferometry [2]

Algorithm 3 Traffic-noise interferometry within HAEE as the user-defined function on DAS data.

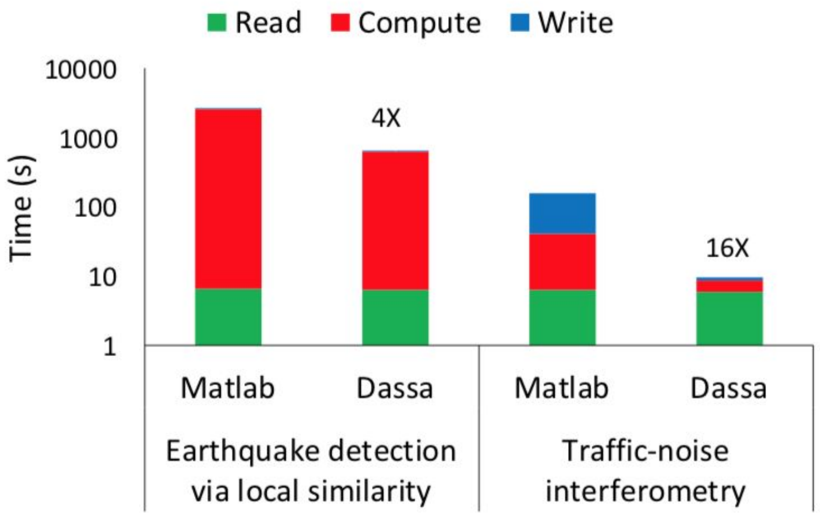
Note: S is the Stencil abstraction representing an abstract cell and its neighborhood. Each channel has a time series of length W. M_{fft} denotes the FFT transformed master channel for each process.

function TRAFFICNOISEUDF(S)
  $W_{n,0} = S(0:(W-1), 0)$ ▷ Time series per channel
  $W_{n,1} = \text{Das\_detrend}(W_{n,0})$
  $W_{n,2} = \text{Das\_filtfilt}(Das\_(n, fc), W_{n,1})$
  $W_{n,3} = \text{Das\_resample}(W_{n,2})$
  $W_{\text{fft}} = \text{Das\_fft}(W_{n,3})$
  return $\text{Das\_abscorr}(W_{\text{fft}}, M_{\text{fft}})$
end function

[2] Ajo-Franklin, J. B., Dou, S., Lindsey, N. J., Monga, I., Tracy, C., Robertson, M., Rodriguez Tribaldos, V., Ulrich, C., Freifeld, B., Daley, T.,and Li, X. (2019) Distributed Acoustic Sensing Using Dark Fiber for Near-Surface Characterization and Broadband Seismic Event Detection, Scientific Reports

Performance Measurements at NERSC[1]

Case Studies: two scientific applications



Matlab

- platform used by current DAS team
- Multithreaded

Test data set

- a single 1-minute file (≈700MB)
- limited by Matlab licence at NERSC

DASSA is up to 16X faster than Matlab

[1]https://www.nersc.gov/

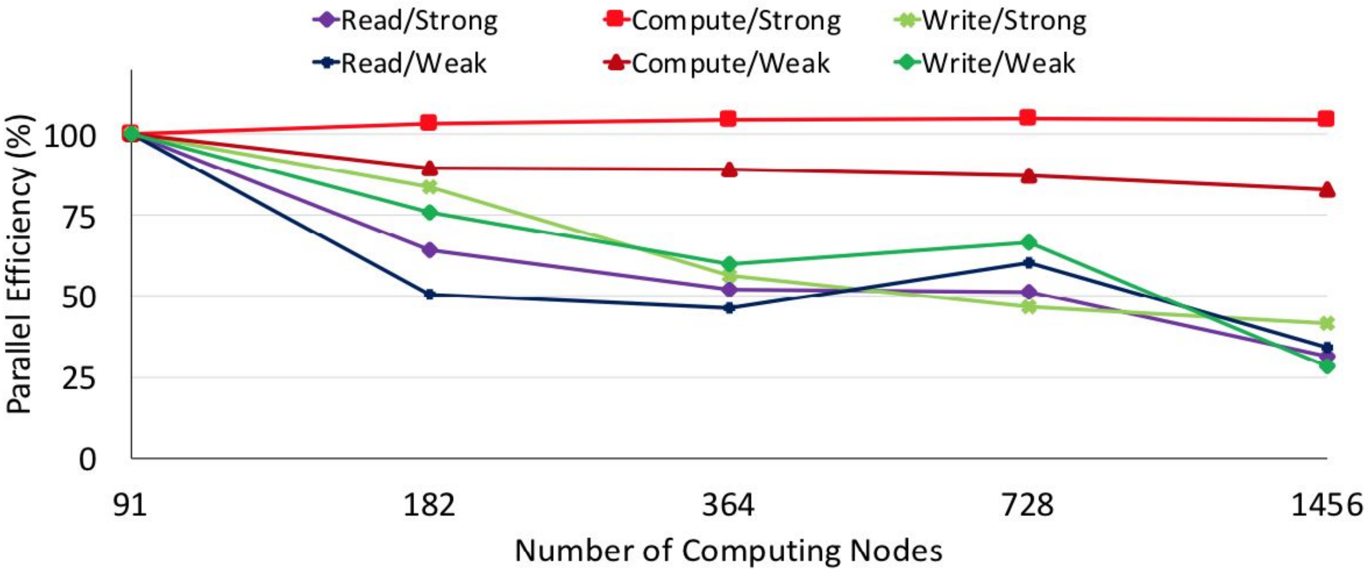
Scalability of DASSA Analysis Engine

Observations about DASSA

- Computing has perfect scaling
- Parallel efficiencies of I/O operations drop below 50% with more than 1000 nodes

Parallel efficiency = $\begin{cases} t_1/t_N * 100\% & \text{Weak Scaling} \\ t_1/(N * t_N) * 100\% & \text{Strong Scaling} \end{cases}$

t_i is the time to finish the work with i CPU cores



Acknowledgement

- U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231

- U.S. Department of Energy (DOE), Office of Energy Efficiency and Renewable Energy (EERE), Office of Technology Development, Geothermal Technologies Office, under contract number DE-AC02-05CH11231

- National Energy Research Scientific Computing Center (NERSC)

Questions ?

Please contact *Bin Dong (dbin@lbl.gov)*