How to Accelerate Backtests By Leveraging the Open Source MPI Framework

June 20, 2017

Gerd Heber, The HDF Group

The HDF Group

Copyright 2017, The HDF Group.

Who is the HDF Group?

HDF Group has developed open source solutions for Big Data challenges for nearly 30 years

Small company (~ 40 employees) with focus on High Performance Computing and Scientific Data

Offices in Champaign, IL + Boulder, CO

Our flagship platform – HDF5 – is at the heart of our open source ecosystem.

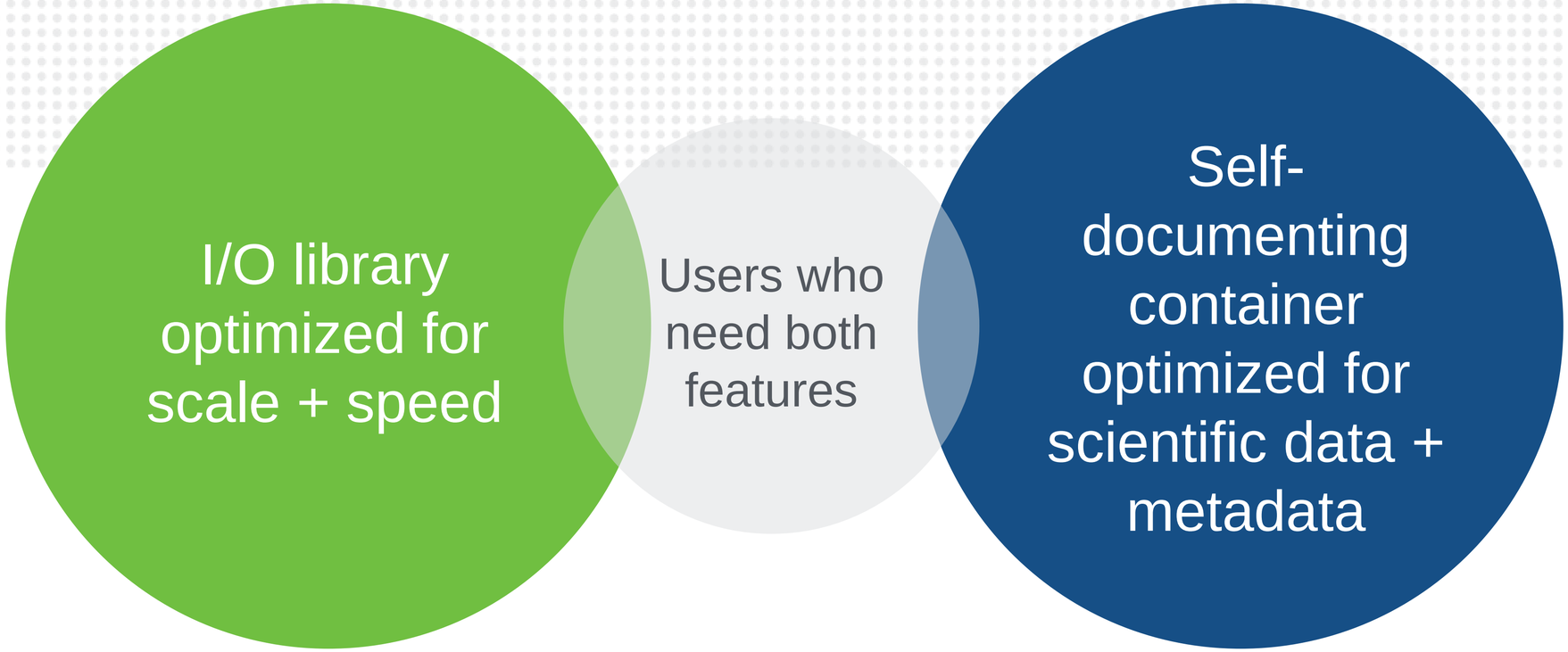
Tens of thousands use HDF5 every day, as well as build their own solutions (~~600~~ ~~700~~ 800+ projects on Github)

“De-facto standard for scientific computing” and integrated into every major analytics + visualization tool

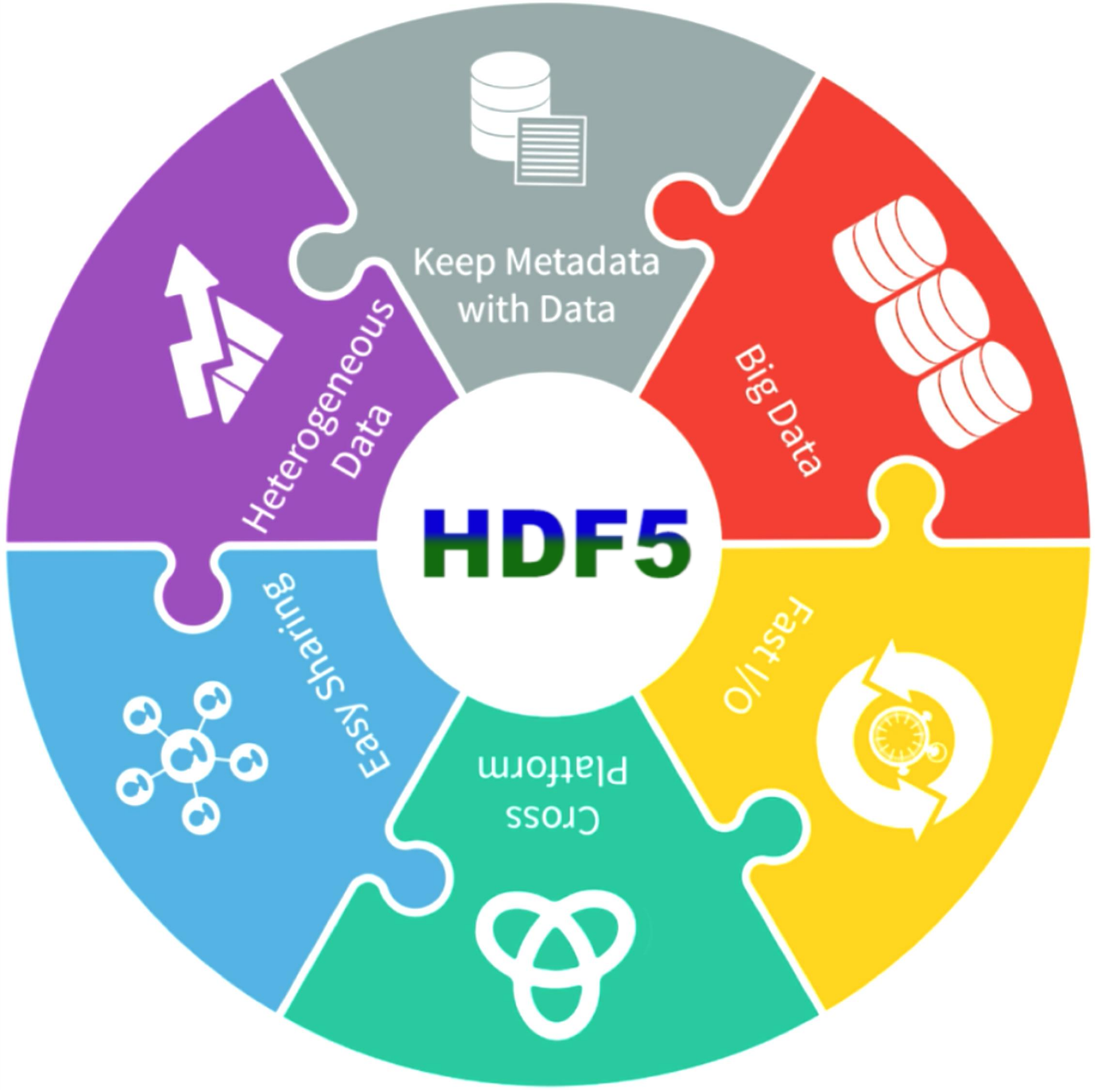
What does the HDF Group offer?

Products	• HDF Capture: Software solution for PCAP Ingest + Storage (Beta) • HDF5 Library • Connectors: ODBC + Cloud (Beta) • Add-Ons: compression + encryption
Support	• HDF Support Packages (Basic + Pro + Premier) • Support for h5py + PyTables + pandas (NEW) • Training
Consulting	• HDF: new functionality + performance tuning for specific platforms • General HPC software engineering with fintech expertise (ex. MPI implementation for back testing) • Metadata science and expert services

Why Use HDF5?

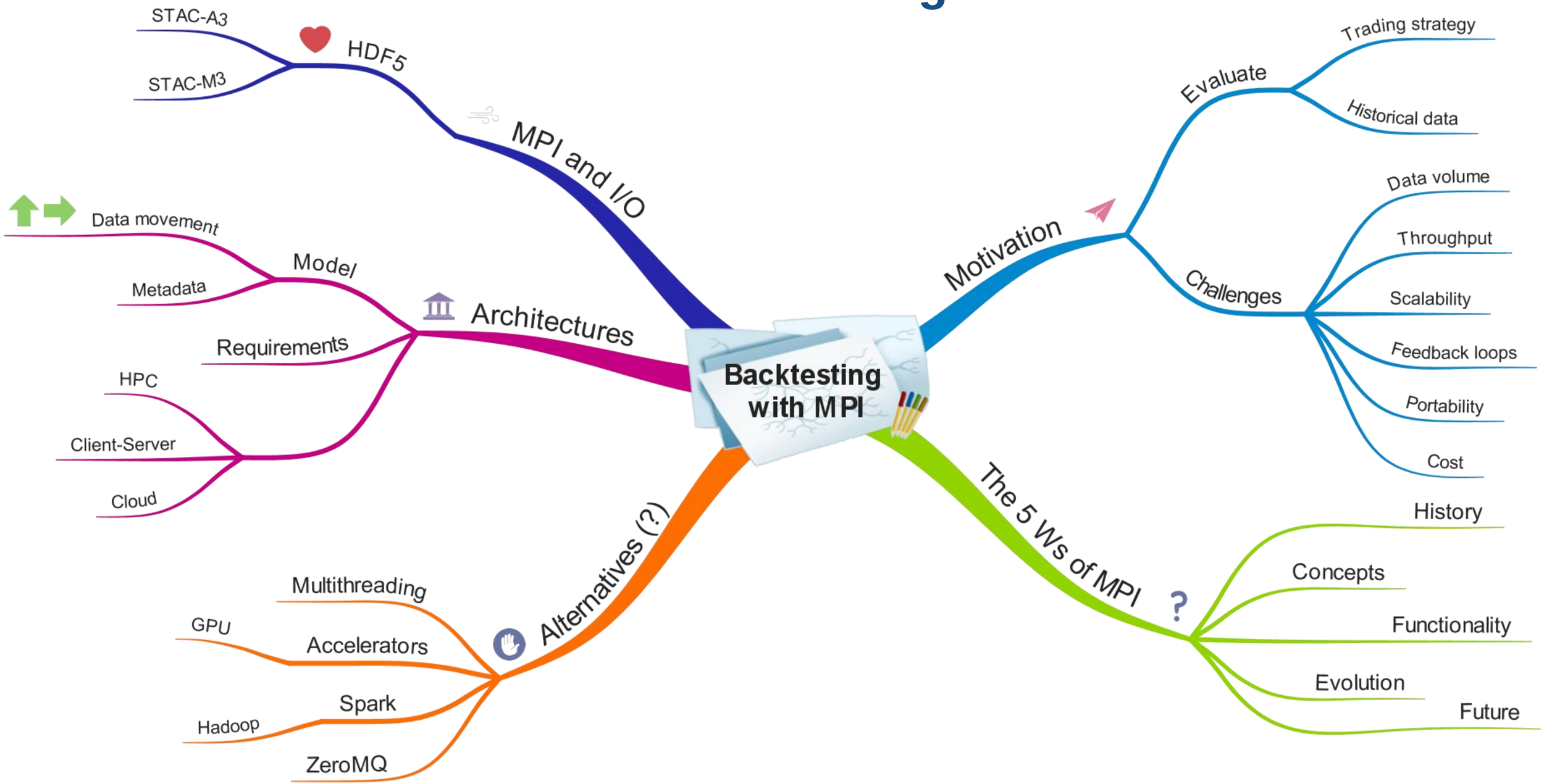

I/O library optimized for scale + speed
Users who need both features
Self-documenting container optimized for scientific data + metadata

Why is this concept so different + useful?



- **Native support for multidimensional data**
- **Data and metadata in one place => streamlines data lifecycle & pipelines**
- Portable, no vendor lock-in
- Maintains logical view while adapting to storage context
- In-memory, over-the-wire, on-disk, parallel FS, object store
- Pluggable filter pipeline for compression, checksum, encryption, etc.
- High-performance I/O
- Large ecosystem (700+ Github projects)

Food for Thought



STAC-M3

- Time-series data
- Software innovations
- I/O-intensive
- Large store of market data
- Agnostic to architecture
- Scaling
- Data volumes
- User (client) counts

https://stacresearch.com/system/files/central/STAC-M3_Overview.pdf

STAC-M3 STAC-M3

STAC BENCHMARK COUNCIL

STAC-M3: The industry standard benchmark suite for tick database stacks

Analyzing time-series data such as tick-by-tick quote and trade histories is crucial to many trading functions, from algorithm development to risk management. Recent trends like the growth and sophistication of automated trading and the proliferation of new regulations place a premium on technology that can accelerate the analysis of time-series data.

In 2010, several of the largest global banks and other trading firms in the STAC Benchmark™ Council joined forces to develop common ways to measure the extent to which emerging hardware and software innovations improve the performance of tick analytics. The result was STAC-M3™.

The STAC-M3 Benchmark suite assesses the ability of a solution stack such as columnar database software, servers, and storage, to perform a variety of I/O-intensive and compute-intensive operations on a large store of market data. The specifications are completely agnostic to architecture, which means that STAC-M3 can be used to compare different products or versions at any layer of the stack, such as database software, processors, memory, hard disks, SSD, interconnects, and file systems.

STAC-M3 consists of a baseline suite that provides performance insight using a modest amount of gear, plus as an optional scaling suite that increases data volumes and simulated user counts.

Dozens of STAC Reports™ have been published using STAC-M3, either publicly or in the members-only STAC Vault™. In addition, numerous user firms, database vendors, and hardware vendors use STAC-M3 to "mark their performance to market" in the privacy of their own labs.

To get acquainted with STAC-M3, read one of the many public reports at www.STACresearch.com/m3. For more information, please contact council@STACresearch.com.

Get the most from STAC-M3

Any interested party can analyze public STAC Reports to compare the performance of different systems. However, members of the STAC Benchmark Council are able to put these reports to much greater use. Qualified members may:

- Read the detailed test specifications
- Access additional reports in the confidential STAC Vault
- Obtain the materials to run the STAC-M3 Benchmarks on their own systems
- Discuss benchmarks, technologies, and related business issues with their peers.

Rev 2

Copyright © 2013 Securities Technology Analysis Center LLC
STAC and all STAC names are trademarks or registered trademarks of STAC.

The HDF Group

Solution = Parallelism + I/O

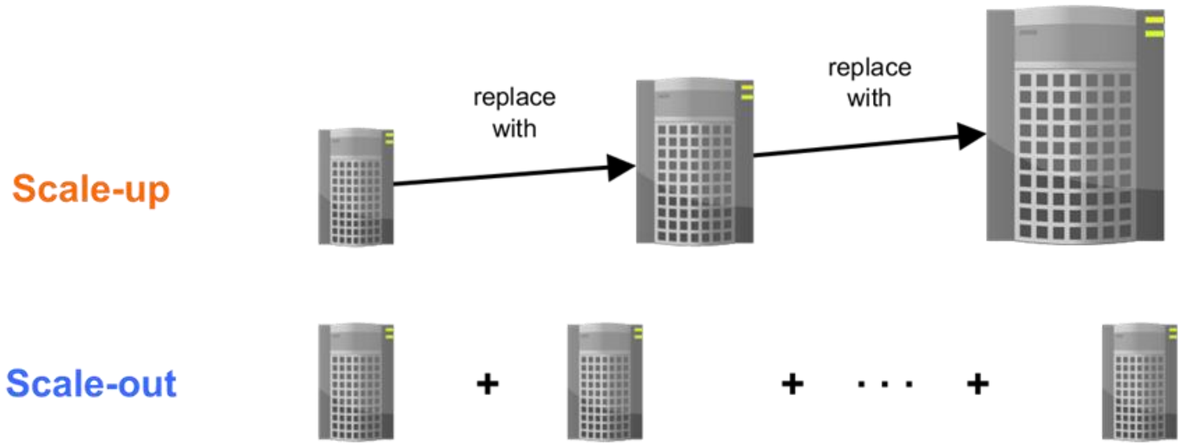
<u>Ingredients</u>:

1. A framework that lets us express parallelism: MPI
2. A storage manager: HDF5

- The combination must scale up and out!

<u>=> Divide & Conquer</u>

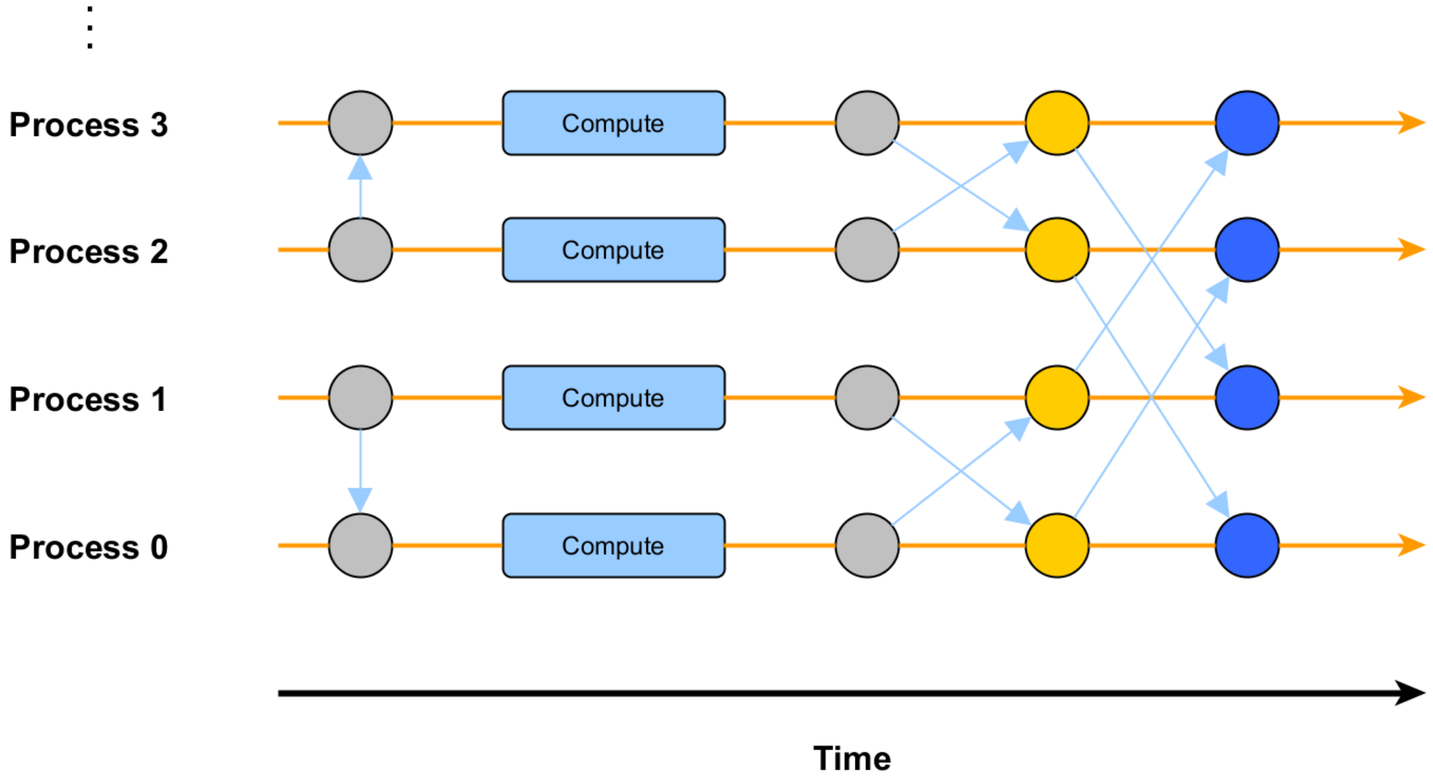
- How do I solve my *one* problem faster?
- How do I do *many* things (= smaller problems) at once?
- How do I deal with potential *interdependence*?



The Message Passing Model

1) Parallel programs consist of *cooperating* processes, each with its *own* memory

2) Processes send data to one another as *messages*



3) Messages may have *tags* that may be used to sort messages

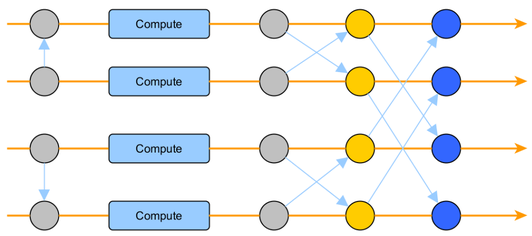
4) Messages may be received in any *order*

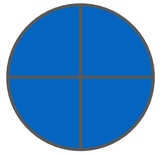
MPI is very simple

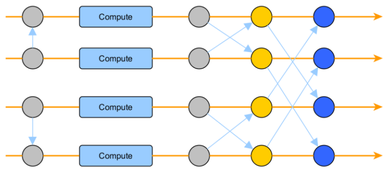
Six functions allow you to write many programs:

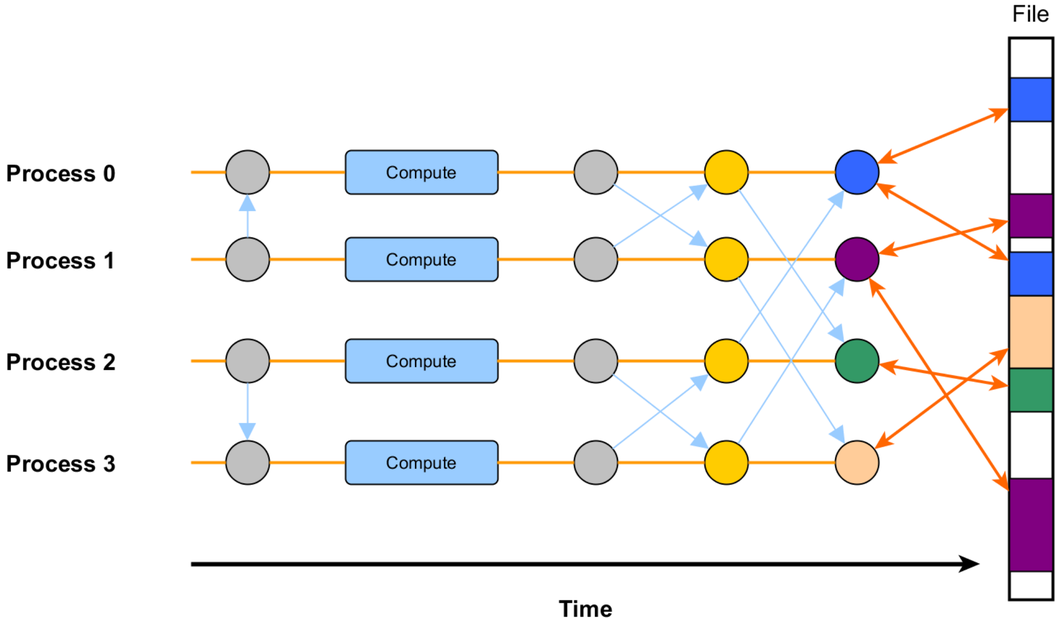
MPI_INIT	Let's get this show on the road!
MPI_FINALIZE	Let's go home!
MPI_COMM_SIZE	How big is the MPI world?
MPI_COMM_RANK	Where am I in the MPI world?
MPI_SEND	Send someone a message
MPI_RECV	Receive someone's message

But wait, there's (a lot) more!









MPI-1 (1994)

- MPI groups, communicators
- P2P + collective communications
- Derived datatypes

MPI-2 (1999)

- Dynamic process management
- Parallel I/O
- One-sided communication

MPI-3 (2012)

…

Is MPI Large* or Small? (Bill Gropp)

- **MPI is large (~ 900 functions as of MPI 3.0)**
  - MPI's extensive functionality requires many functions
  - Number of functions not necessarily a measure of complexity
- **MPI is small (6 functions)**
  - Many parallel programs can be written with just 6 basic functions.
- **MPI is just right**
  - One can access flexibility when it is required.
  - One need not master all parts of MPI to use it.
- **Ditto for HDF5 (~ 500 functions as of HDF5 1.10)**

*ANSI Common Lisp has 978 symbols.

SQL

ANSI SQL has about 825 reserved words.

ISO C++ has about 100 reserved words.

The 5 Ws of MPI

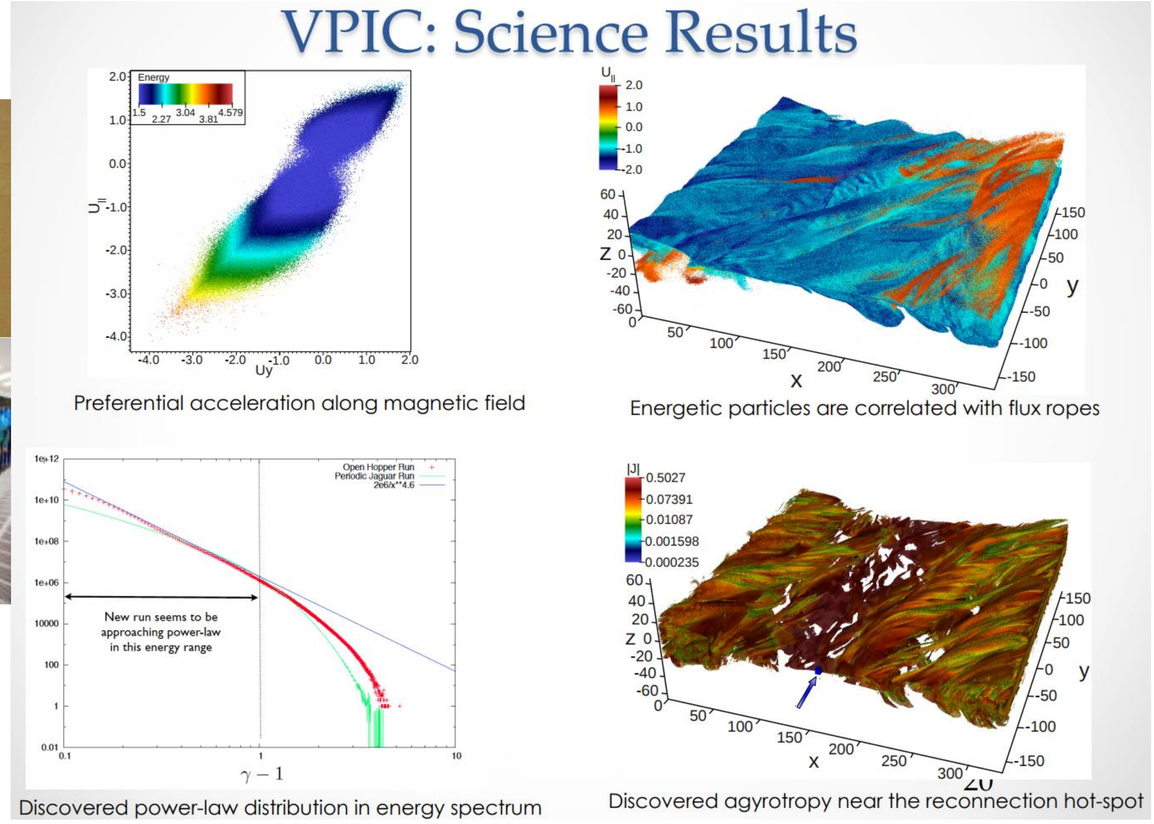
25 YEARS OF IMPACT 1992 • 2017

The HDF Group

- **M**essage **P**assing **I**nterface specification
- ~~Since before you were born~~ Early 90s (04/92)
- Goals: Portability, performance, openness
- De-facto standard for large-scale parallel apps.
- C/C++, Fortran, Java, Python language bindings
- Standards body: MPI Forum
- Current standard version MPI 3.1 (2015)
- Talent pool: Do[E,D] (U.S.), EPSRC, Met Office (U.K.)

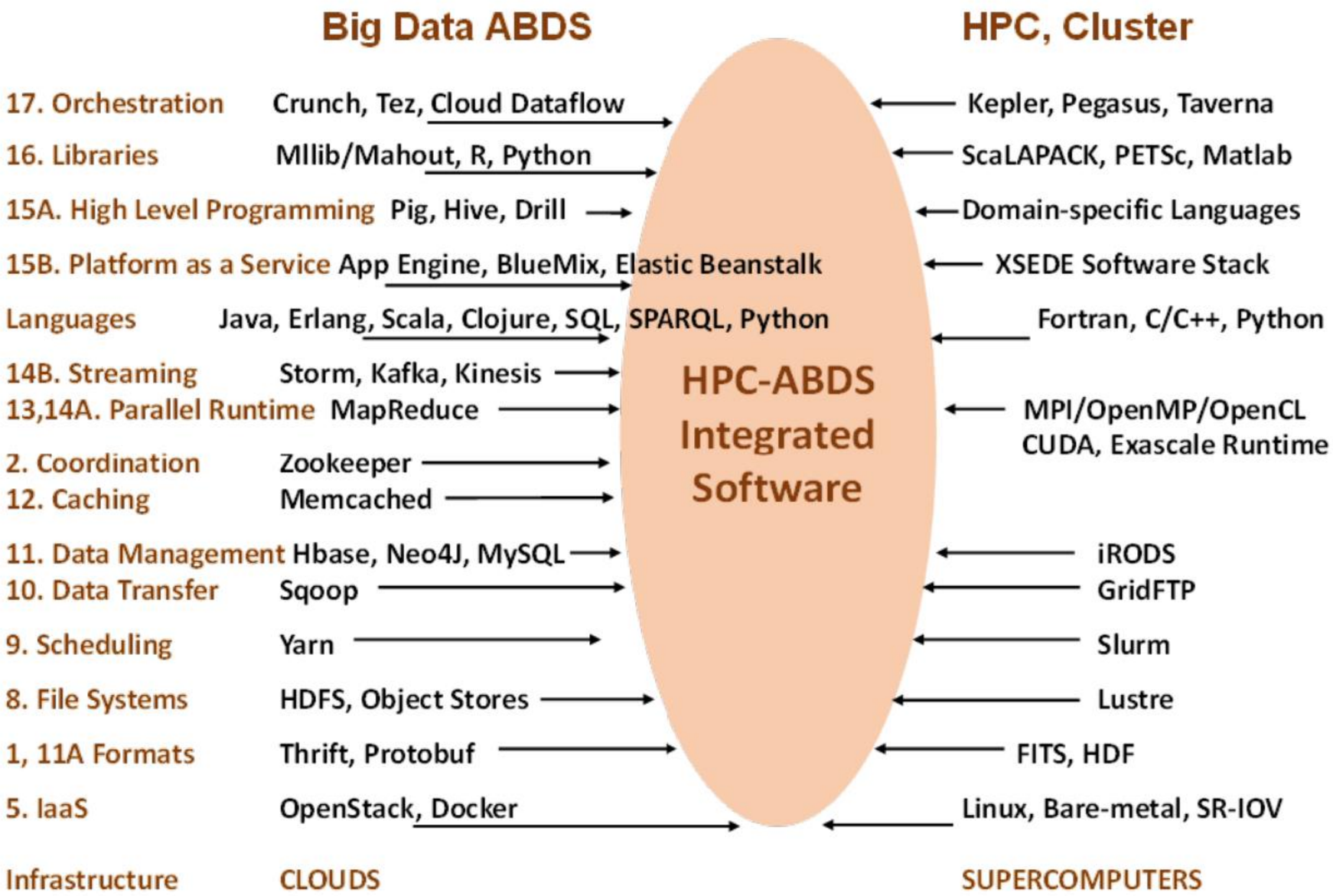
“MPI goes to eleven.” (Nigel Tufnel)

- Byna et al. (2013) on a Cray XE6 @ NERSC
- A trillion particle plasma
- 120,000 cores
- 30 - 43 TB per timestep
- 10,000 timesteps simulated
- 100 timesteps dumped ~350 TB
- 150 TB checkpoint data
- Sustained 35 GB/s write bandwidth (80% of peak) to a <u>single</u> HDF5 file in a Lustre parallel FS



"A supercomputer is a device for turning compute-bound problems into I/O-bound problems." (Ken Batcher)

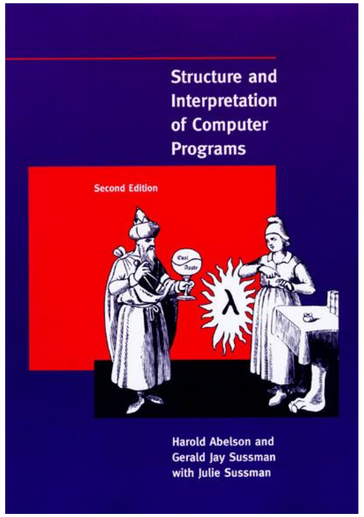
Evolution & Future – Convergence?



Source: Fox et al., *High-Performance Computing Enhanced Apache Big Data Stack*, IEEE/ACM CCGrid 2015.

Divergence?

"A grand unification has yet to emerge."
(Sec. 3.5.5, 1984)



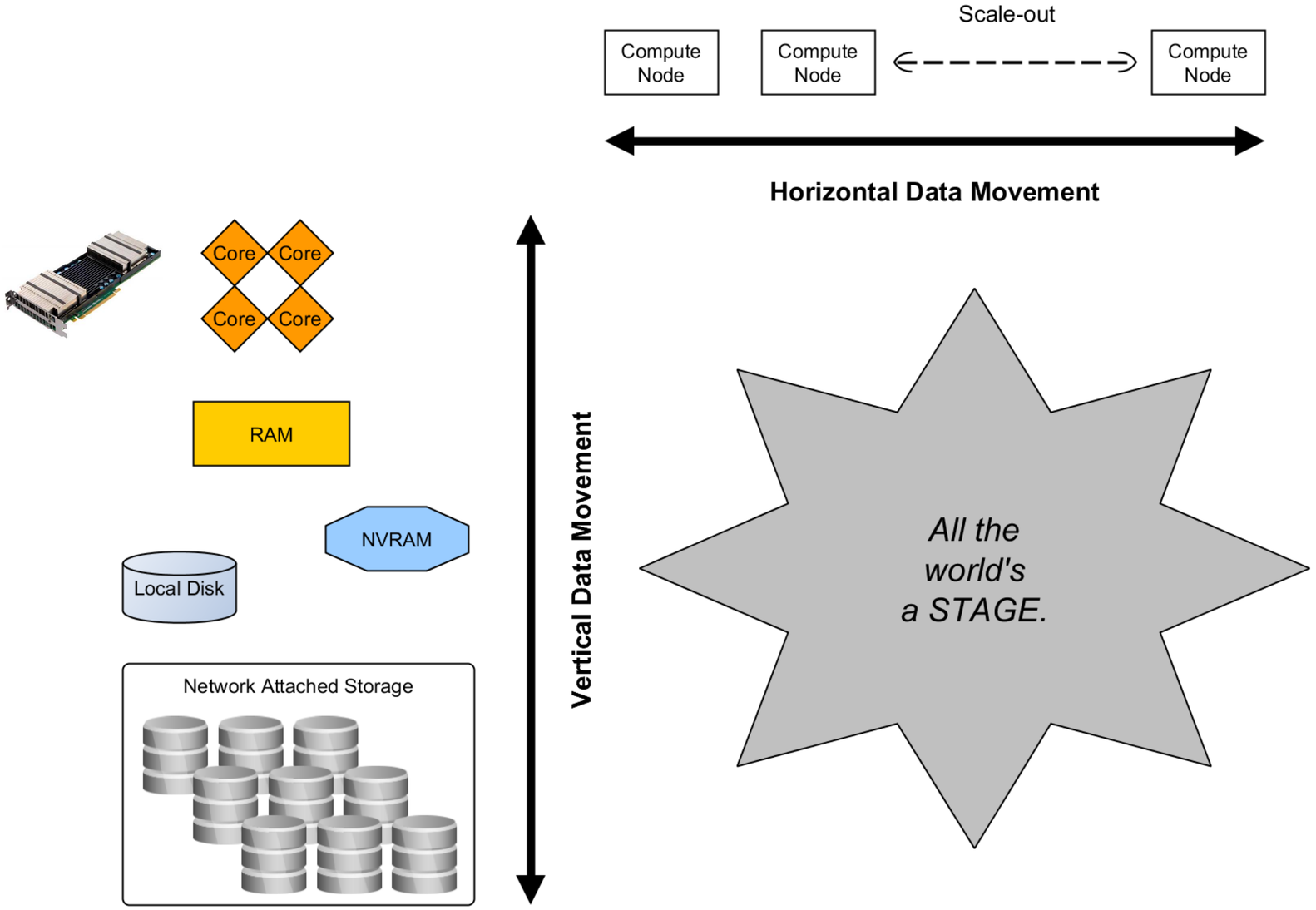
MPI

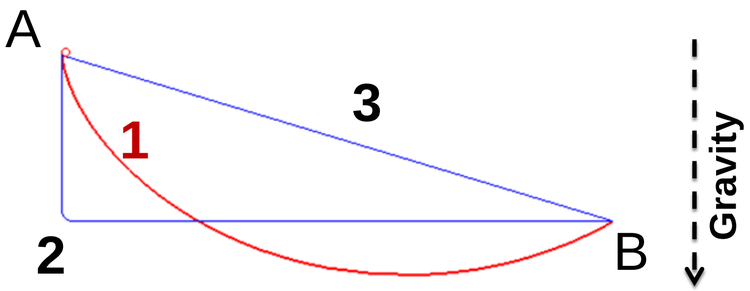
- In-memory, in-place execution
- Stateful computations
- Resource efficient
- Arbitrary control flows
- P2P and collective communications
- Can be complex

Data Flow (Spark, Flink, …)

- Stateless computations
- Functional programming
- Resource hungry
- Simple control flows
- No inter-task communication
- Less error prone

Architectural Elements - 1





Curve of fastest descent
(Johann Bernoulli, 1697)

Architectural Elements - 2

Big Data

- Local storage
- Low-latency vertical data movement
- High-latency interconnect
- BW ~ # compute nodes
- WORM file access
- COTS hardware
- Fault tolerance

High-Performance Computing

- “Centralized” || file system(s)
- High-latency vertical data movement
- Low-latency interconnect
- Parallel access paths
- Many-write-many-read
- Non-commodity hardware
- Global namespace

Source: N. Chaimov et al., *Scaling Spark on HPC Systems*, Proceedings ACM HPDC'16, 2016.

Alternatives - Metrics

+ : good ☺
* : fair 😐
- : poor ☹

The HDF Group

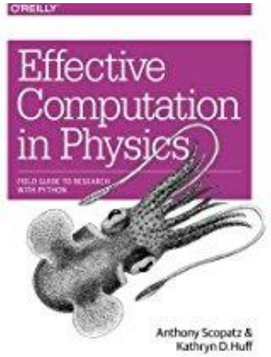
Metric	XYZ	MPI
Fault tolerance		- (*) Not built-in, left to the user
Collectives		+ Fully supported, highly optimized
Dynamic resources		- (*) Not built-in, supported in some frameworks
Communication protocols		+ The faster the better
Level of abstraction		- + Low & high, generally below other frameworks
Resource usage (CPU, RAM, I/O)		+ Very low overhead
Development time		- Relatively higher, esp., performance features
Expertise required		- Relatively higher
Time to solution		+ Typically much (10+ x) faster
Portability		+ Core value
Scale-up		+ Typically better than SMP models
Scale-out		+ Excellent

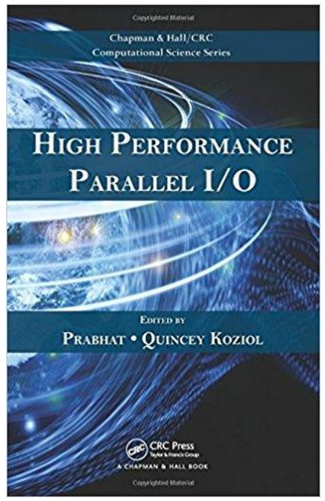
Source: S. Kamburugamuve et al., *Anatomy of machine learning algorithm implementations in MPI, Spark, and Flink.* Technical Report, 2017.

Implementation Strategy

STAC-M3/A3

- Pleasingly data parallel (symbols, dates, …)
- Modest compute per bytes read/written
- I/O bound





MPI

- Think MapReduce+++, not Client/Server
- You partition the workload
- Assess data access path
- Balance vertical & horizontal data movement
- Scalability through symmetry ~~primary/replica~~
- Prototype in Python w/ `mpi4py` & `h5py` or `HPAT*`
- C/C++ for the latest (HDF5) features

*Totoni et al. (2017), *HPAT: High Performance Analytics with Scripting Ease-of-Use*, arXiv:1611.04934

Template

Simple Control Flow

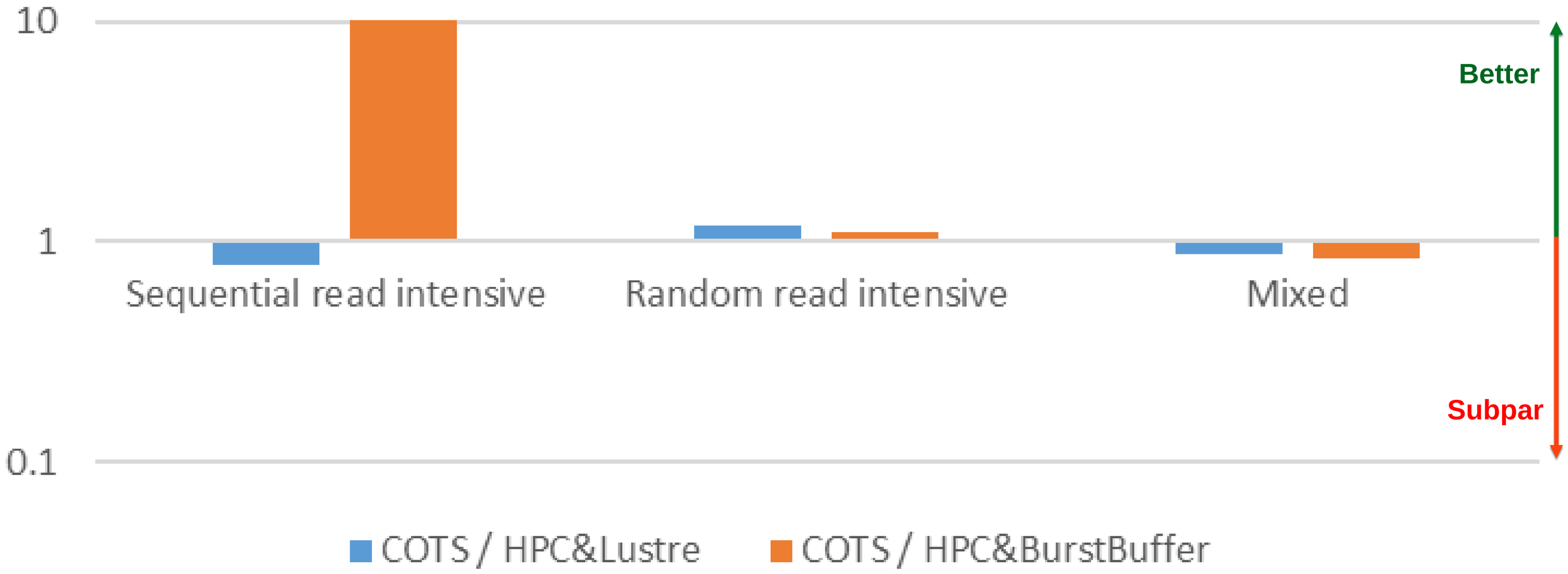
1. Read RRD*
2. Partition RRD
3. Read tick data
4. Compute
5. Buffer/write results

*RRD – Randomized Reference Data: query parameters such as dates, symbols, exchanges

(Too?) Many Options

- Distribution across files (e.g., file per symbol/year)
- Layout / compression (HDF5: 170-450% storage eff.)
- Metadata (Indexing, look-up tables)
- N-to-M reads and writes
- A mixture of vertical and horizontal data movement
- Read/write aggregation

Sample Tickdata Operations – COTS vs. HPC



This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility Supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Parting Words

- MPI and HDF5 are a mature (20 years+) couple
- Shared values: portability, performance, openness
- Oldies? At the forefront of Exascale computing!
- Fancy hardware helps but is not mandatory
- MPI + HDF5 delivers scalable backtest performance (scale-out and scale-up!)
- *You don’t need to write MPI code. Compilers (e.g., HPAT) can generate it for you.*
- Helps you to keep an open mind about the future
- Remember:

Lifetime of code >> machine

Questions? Comments?

www.hdfgroup.org

Dave Pearah
CEO
David.Pearah@hdfgroup.org

Dax Rodriguez
Director of Commercial Services and Solutions
Dax.Rodriguez@hdfgroup.org