Universidad Complutense de Madrid
Conferencias de Postgrado

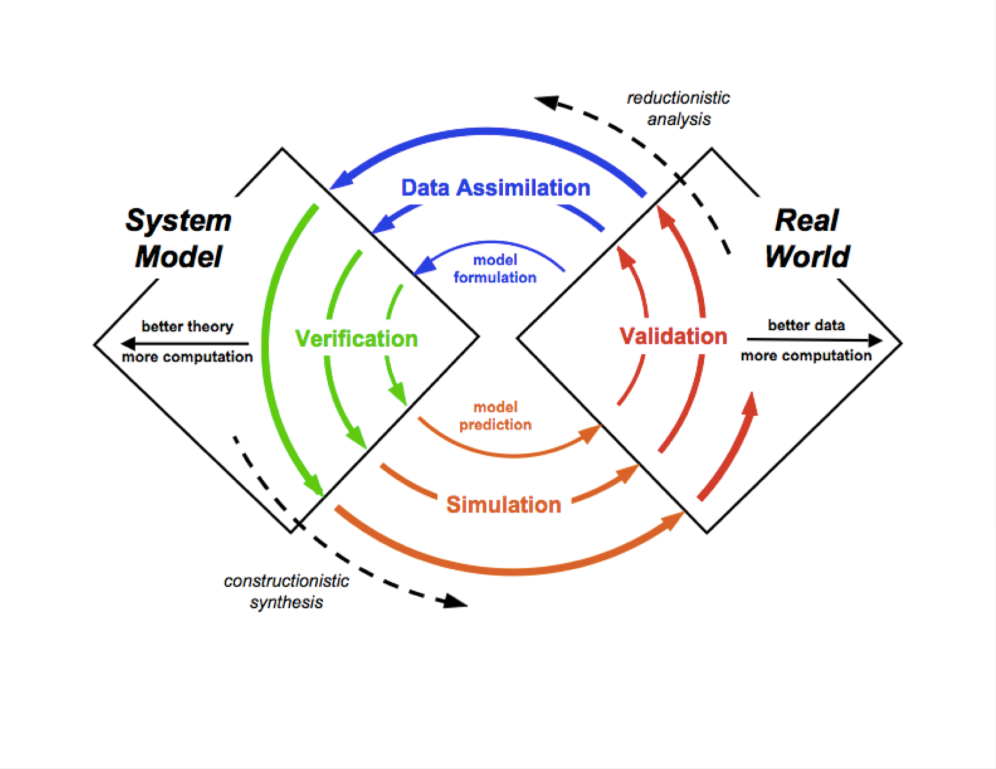
Towards Unification of HPC and Big Data Paradigms

Jesús Carretero

Computer Science and Engineering Department
University Carlos III of Madrid
`jcarrete@inf.uc3m.es`

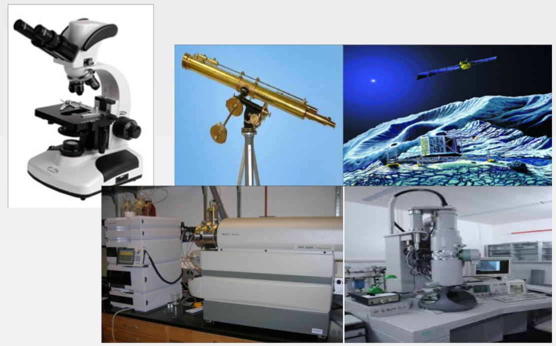
Science research is changing

- Inference Spiral of System Science
  - As models become more complex and new data bring in more information, we require ever increasing computational resources



Who is generating Big Data

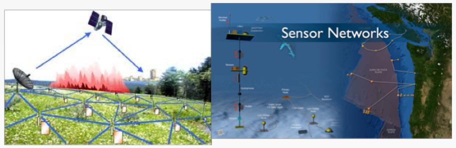
Social media and networks
(all of us are generating data)



Scientific instruments
(collecting all sorts of data)

Mobile devices
(tracking all objects all the time)

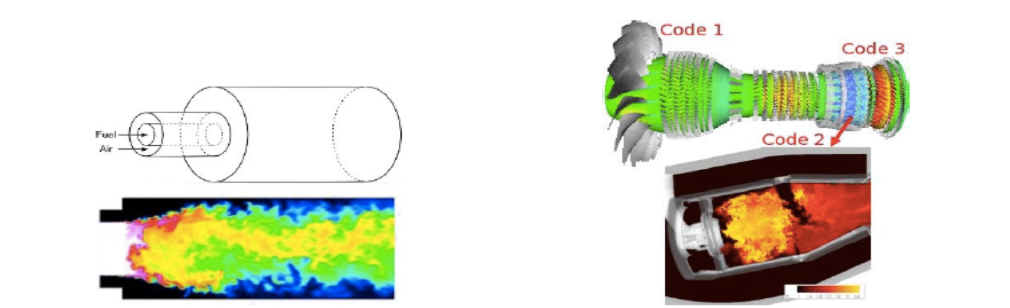
Companies and e-commerce
(Collecting and warehousing data)



Sensor technology and networks
(measuring all kinds of data)

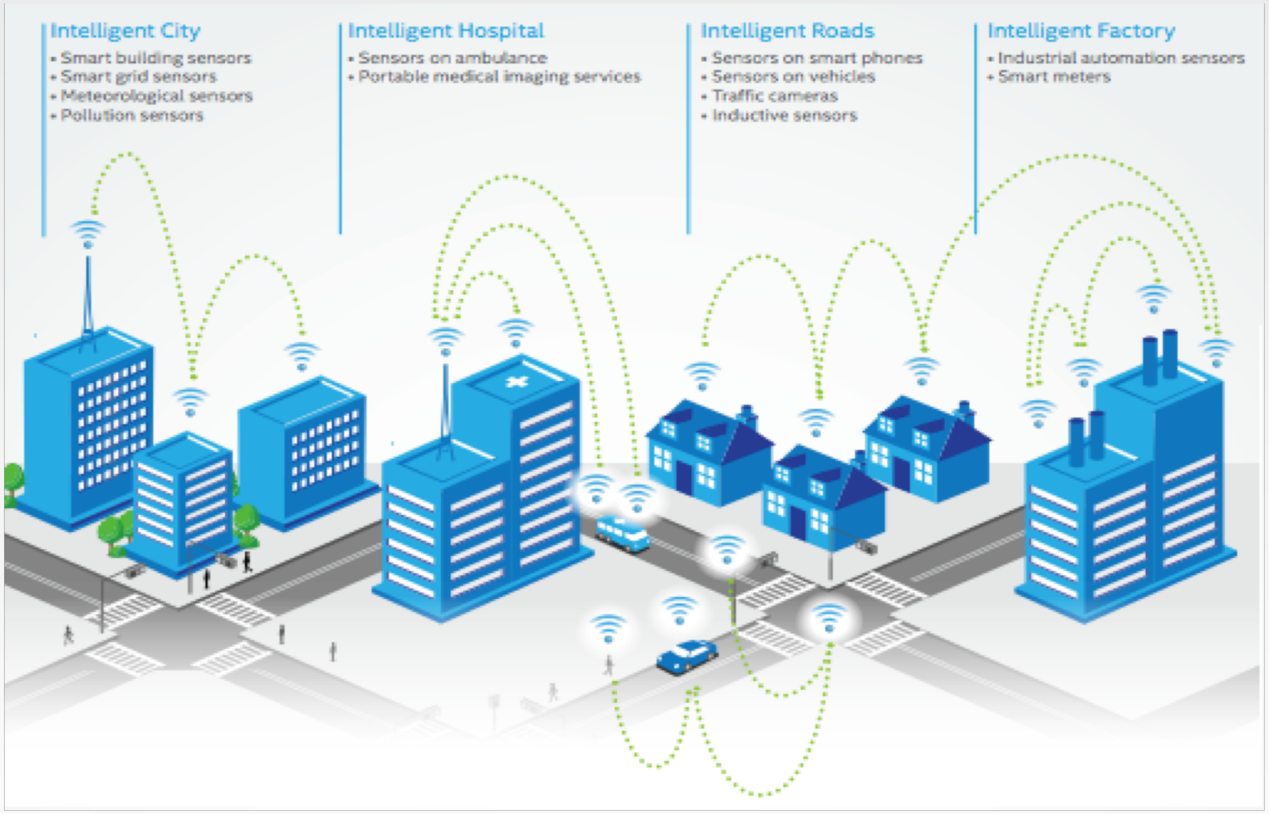
Parallel applications require more data everyday …

- Simulation has become the way to research and develop new scientific and engineering solutions.
  - Used nowadays in leading science domains like aerospace industry, astrophysics, etc.
- Challenges related to the complexity, scalability and data production of the simulators arise.
- Impact on the relaying IT infrastructure.



(a) State of the art in 1997
(b) State of the art in 2007

IoT: the paradigmatic challenge



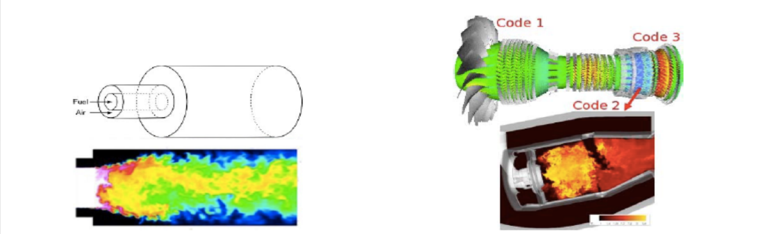
- The progress and innovation is no longer hindered by the ability to collect data
- But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

Cross fertilization needed

- Cross fertilization among High Performance Computing (HPC), large scale distributed systems, and big data management is needed.
  - Mechanisms should be valid for HPC, HTC and workflows …
- Data will play an increasingly substantial role in the near future
  - Huge amounts of data produced by real-world devices, applications and systems (checkpoint, monitoring, …)

Areas of convergence

- HPC Simulations and data
  - Challenges related to the complexity, scalability and data production of the simulators arise.



(a) State of the art in 1997 (b) State of the art in 2007

- High-Performance data analytics (HPDA)
  - More input data (ingestion)
  - More output data for integration/analysis
  - Real time, near-real time requirements

HPC-BD convergence motivation

- Systems are expensive and not integrating misses opportunities
  - Leveraging investments and purchasing power
- Integration of Computation and Observation cycles implicitly requires convergence
- Expanded cross disciplinary teams of researchers are needed to explore the most challenging problems for society
- Data Consolidation trends span Big Data and HPC
  - Categorization of Data
  - Structured, Semi-structured and Unstructured Data
  - Computer Generated and Observed Data

Context

HPC

- **Focus**: CPU-intensive tightly-coupled applications
- **Architecture**: compute and storage are decoupled, high-speed interconnections.

BIG DATA

- **Focus**: large volumes of loosely-coupled tasks.
- **Architecture**: co-located computation and data, elasticity is required.

HPC-Big Data convergence is a must

- Data-intensive scientic computing
- High-performance data analytics
- Convergence at the infrastructure layer
  - virtualisation for HPC, deeper storage hierarchy, …

HPC and Big Data models

- HPC requires Computing-Centric Models (CCM)
- Big Data requires Data-Centric Models (DCM)

BIG DATA PARADIGM (SPARK, HADOOP)	
Pros	**Fault-tolerance by design**
	Transparent data-locality
	Job and task scheduling at platform level
Cons	Low resource management control
	Significant memory overhead
	Poor integration with kernels
	Key-value only
	Deep software and communication stack
HPC PARADIGM (MPI)	
Pros	**Low resource consumption**
	Efficient communication
	Generalist and tailorable processes
Cons	Limited parallel abstractions
	No native provenance or replication

Platforms & paradigms

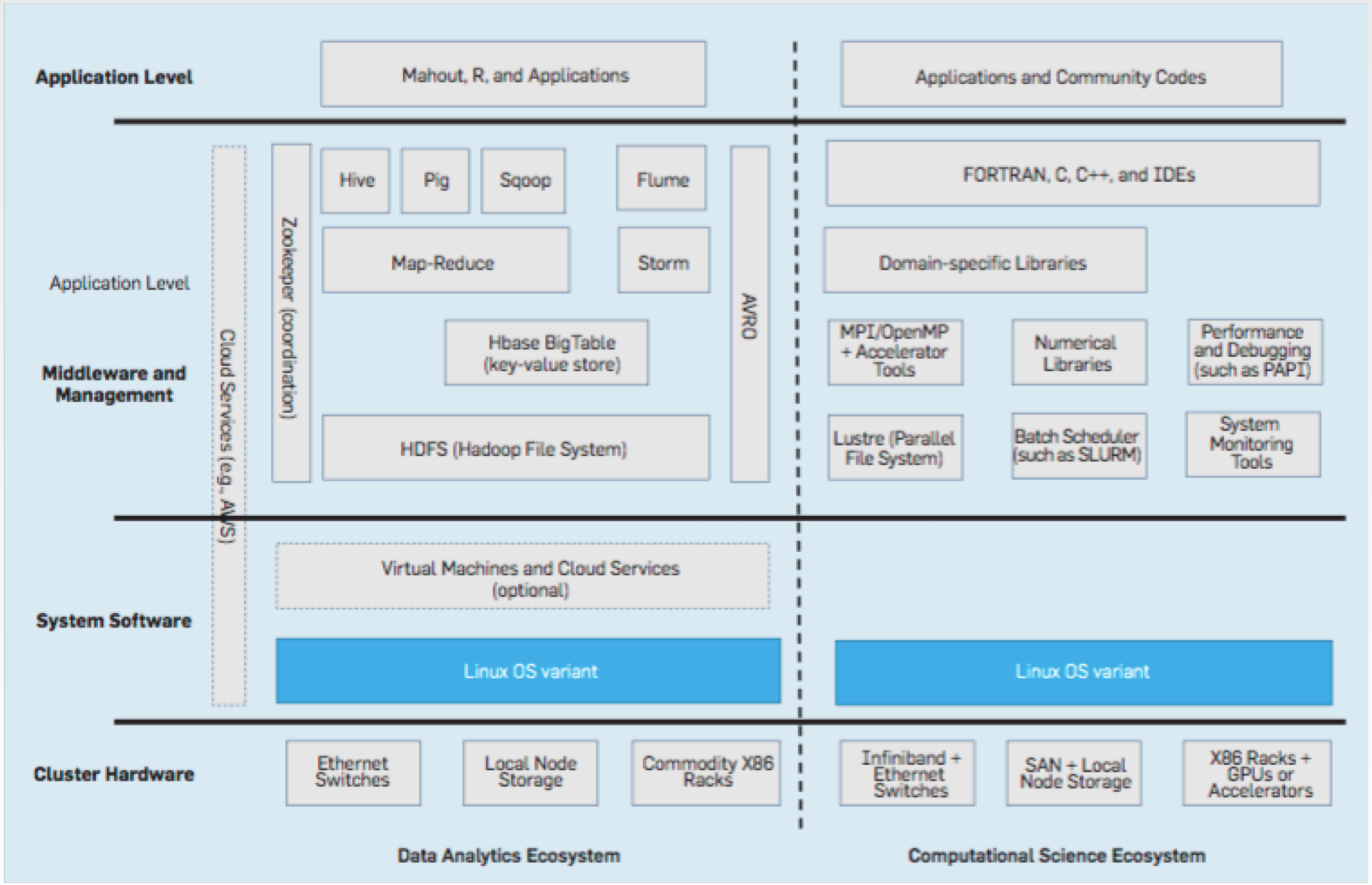
Physical or virtual

- Clusters and supercomputers
  - HPC and supercomputing
- Clouds
  - Virtualized resources
  - Higher-level model

General or specific

- Processing paradigms
  - Open MP and MPI
  - Collective model (PGAs,...)
  - MapReduce model
  - Iterative MapReduce model
  - DAG model
  - Graph model

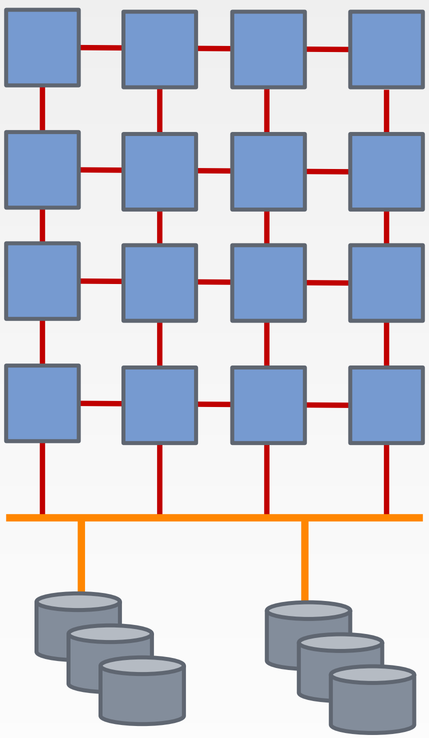
Data analytics and computing ecosystem compared



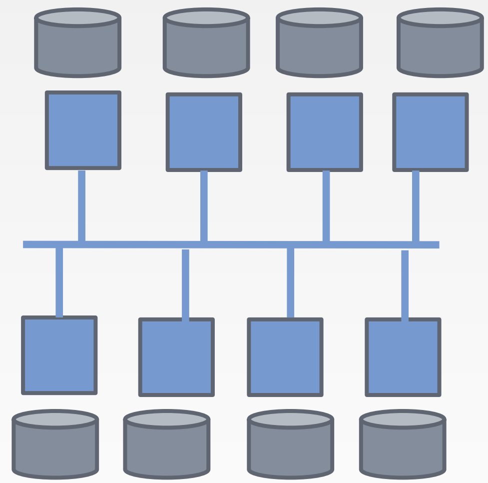
Daniel A. Reed And Jack Dongarra. Exascale Computing and Big Data.Communications Of The Acm. 58(1). July 2015. 7

Non-Convergent system architectures

- HPC system
- Big Data Platforms



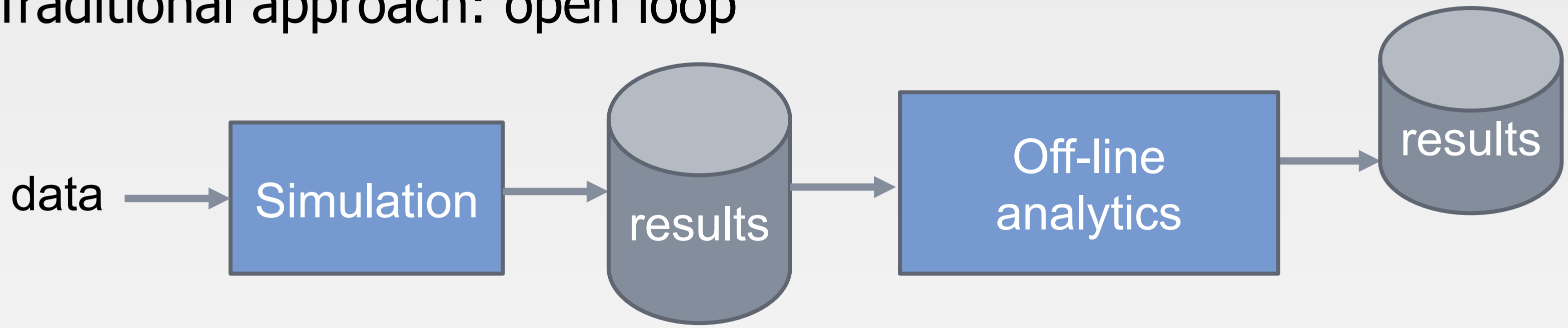
Physical resources



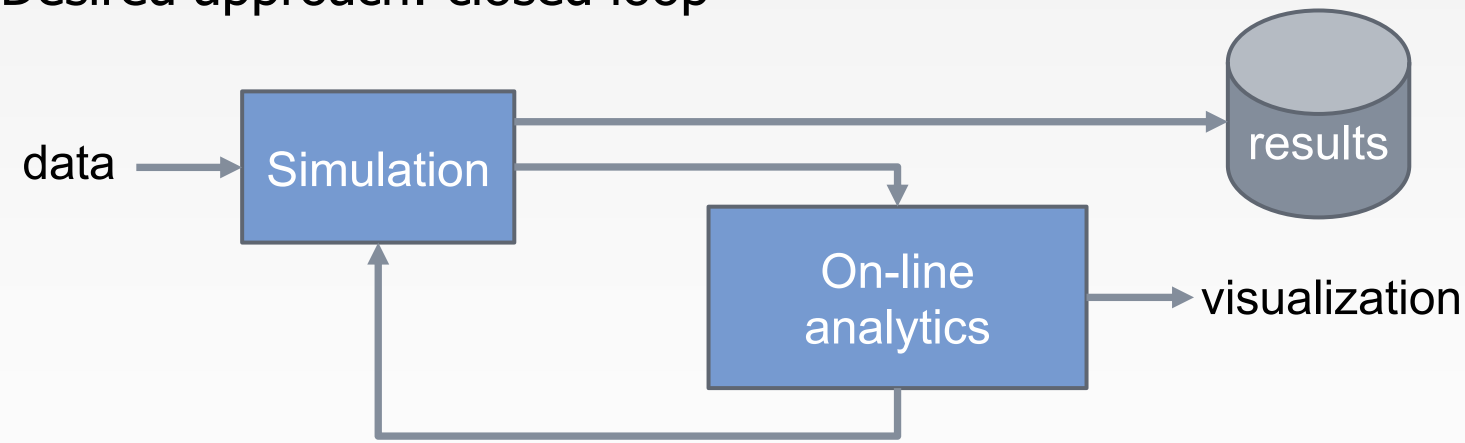
Virtualized resources

Integration of computation and observation

- Traditional approach: open loop



- Desired approach: closed loop



But we need to …

- Integrate the platform layer and data abstractions for both HPC and Big Data platforms
  - We can use Mpi-based MapReduce, but we loose all BD existing facilities.
  - Solution: Connection of MPI applications and Spark.
- Avoid data copies between simulation and analysis every iteration.
  - HPC and BigData use different file systems
  - Copying data will lead to poor performance and huge storage space
  - Solution: Scalable I/O system architecture.
- Have data-aware allocation of tasks in HPC.
  - Schedulers are CPU oriented
  - Solution: connecting scheduler with data allocation.

Convergence in programming environments?

- HPC and BD have separate computing environment heritages.
  - Data: R, Python, Hadoop, MAHOUT, MLLIB, SPARK
  - HPC: Fortran, C, C++, BLAS, LAPACK, HSL, PETSc, Trilinos.

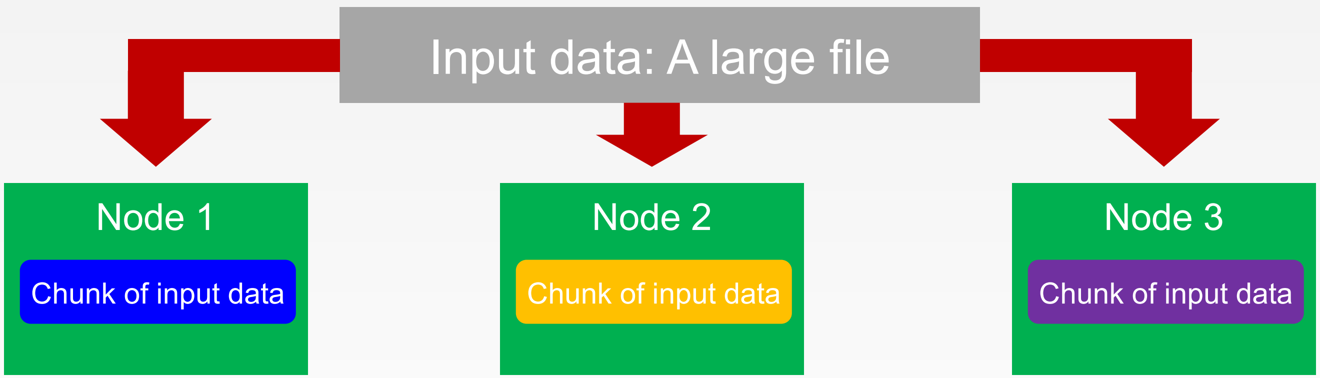
- Determine capabilities, requirements (application, system, user), opportunities and gaps for:
  - Leveraging HPC library capabilities in BD (e.g., scalable solvers).
  - Providing algorithms in native BD environments.
  - Providing HPC apps, libraries as appliances (containers aaS).

MapReduce is the leading paradigm

- A simple programming model
  - Functional model
    - A combination of the Map and Reduce models with an associated implementation
- For large-scale data processing
  - Exploits large set of commodity computers
  - Executes process in distributed manner
  - Offers high availability
  - Used for processing and generating large data sets

Data-driven distribution

- In a MapReduce cluster, data is distributed to all the nodes of the cluster as it is being loaded in.
- An underlying distributed file systems (HDFS) splits large data files into chunks which are managed by different nodes in the cluster



- Even though the file chunks are distributed across several machines, they form *a single namespace (key, value)*
- Scale: Large number of commodity hardware disks: say, 1000 disks 1TB each

Classes of problems "mapreducable"

- Benchmark for comparing: Jim Gray's challenge on data-intensive computing. Ex: "Sort"
- Google uses it (we think) for wordcount, adwords, pagerank, indexing data.
- Simple algorithms such as grep, text-indexing, reverse indexing
- Bayesian classification: data mining domain
- Facebook uses it for various operations: demographics
- Financial services use it for analytics
- Astronomy: Gaussian analysis for locating extra-terrestrial objects.
- Expected to play a critical role in semantic web and web3.0

Data-centric adaptation

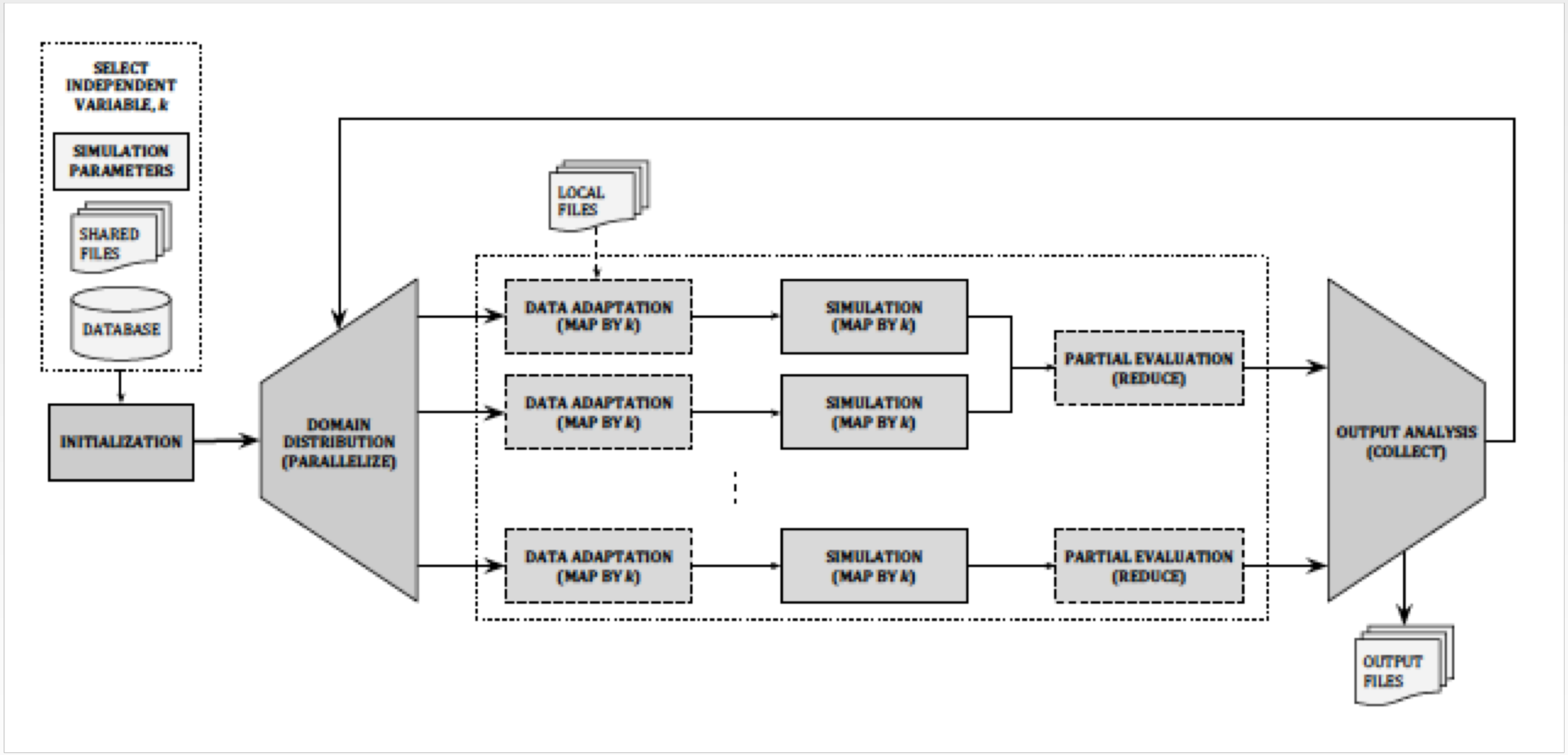
- Find the way to divide the original simulation
  - into smaller independent simulations (BSP model)
- Analyse the original simulation domain in order to find an independent variable Tx that can act as index for the partitioned input data.
  - Independent time-domain steps
  - Spatial divisions
  - Range of simulation parameters

The goal is to run the same simulation kernel but on fragments of the full partitioned data set

Methodology: two phase approach

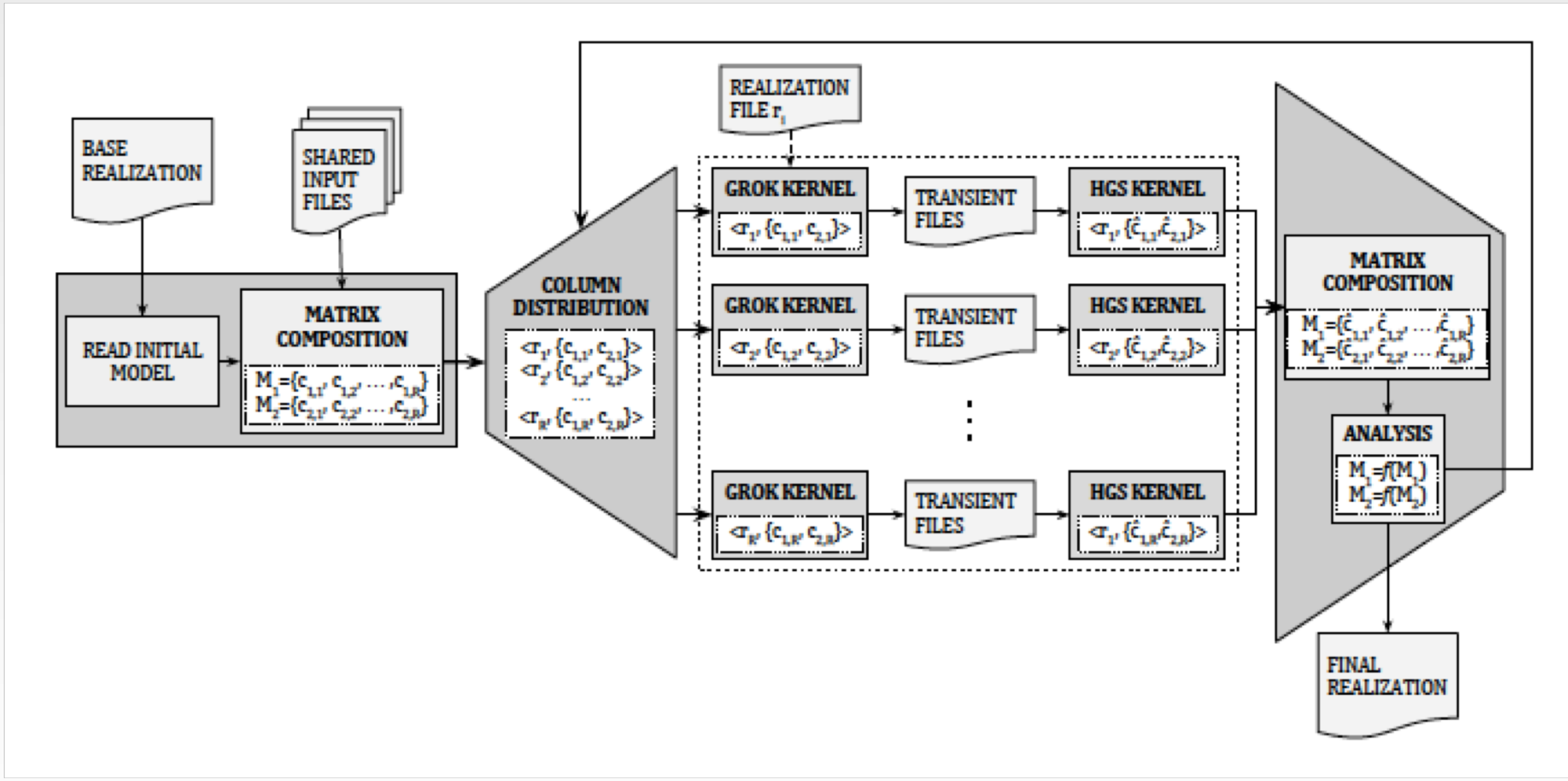
- Data adaptation phase: first Map-Reduce task
  - Reads the input files and indexes all the necessary parameters by Tx
    - Reducers provide intermediate <key, value> output for next step
  - The original data is partitioned
    - Subsequent simulations can run autonomously for each (Tx; parameters) entry.
- Simulation phase: second Map-Reduce task
  - Runs the simulation kernel for each value of the independent variable
    - With the necessary data that was mapped to them in the previous stage
    - Plus the required simulation parameters that are common for every partition
  - Reducers are able to gather all the output and provide final results as the original application.

Data-driven architectural model



"Efficient design assessment in the railway electric infrastructure domain using cloud computing", S. Caíno-Lores, A. García, F. García-Carballeira, J. Carretero, *Integrated Computer-Aided Engineng*, vol. 24, no. 1, pp. 57-72, December, 2016.

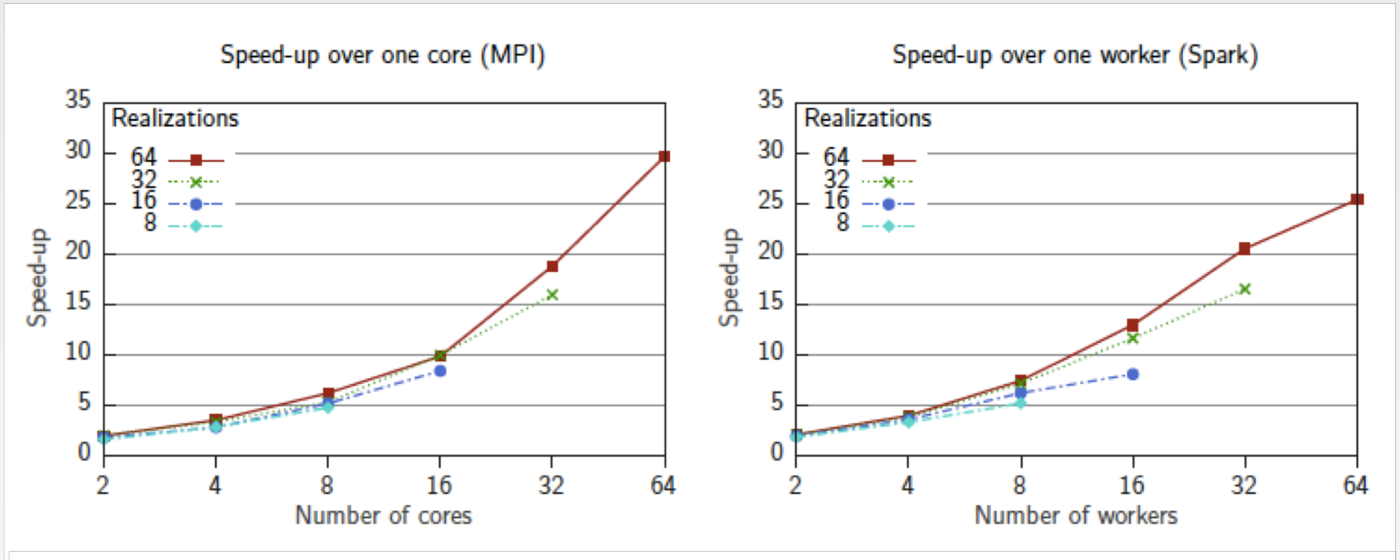
Hydrogeology simulator adaptation



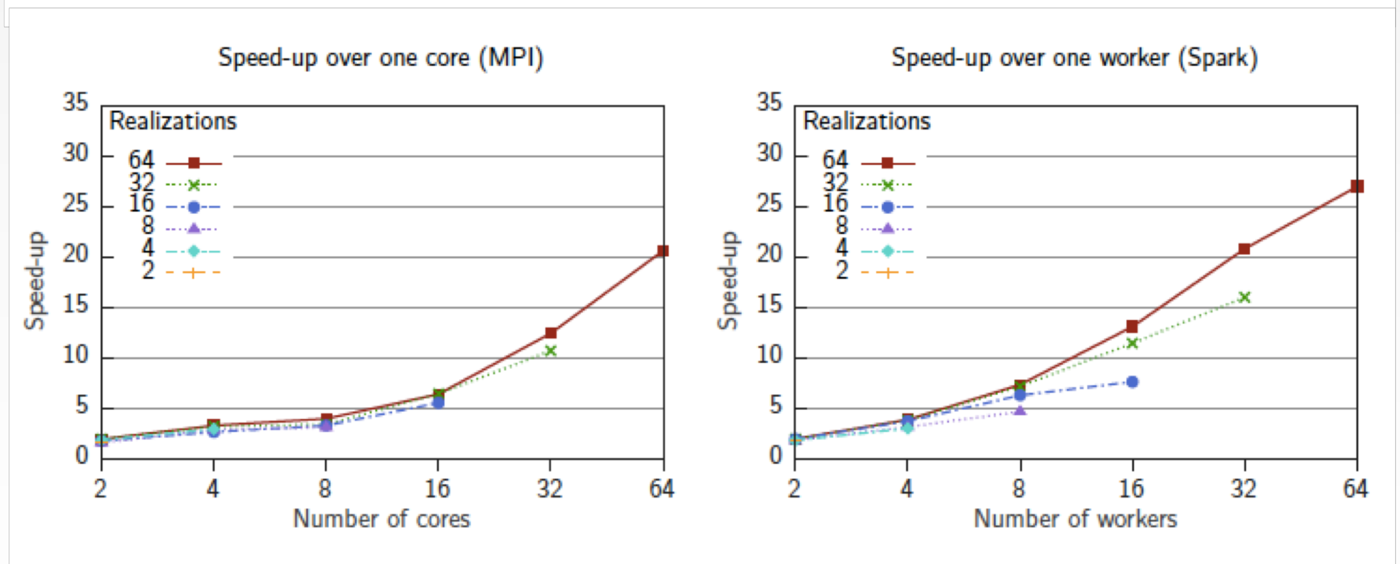
- The ensemble of realizations constitute the parallelizable domain (i.e. key).
- Columns of the model are distributed per realization.

Problem: Scalability

Cluster



EC2



MapReduce framework for MPI

- **MR-MPI**
  - Open-source implementation of MapReduce written for distributed-memory parallel machines on top of standard MPI message passing.
  - C++ and C interfaces and a Python wrapper

- http://mapreduce.sandia.gov/

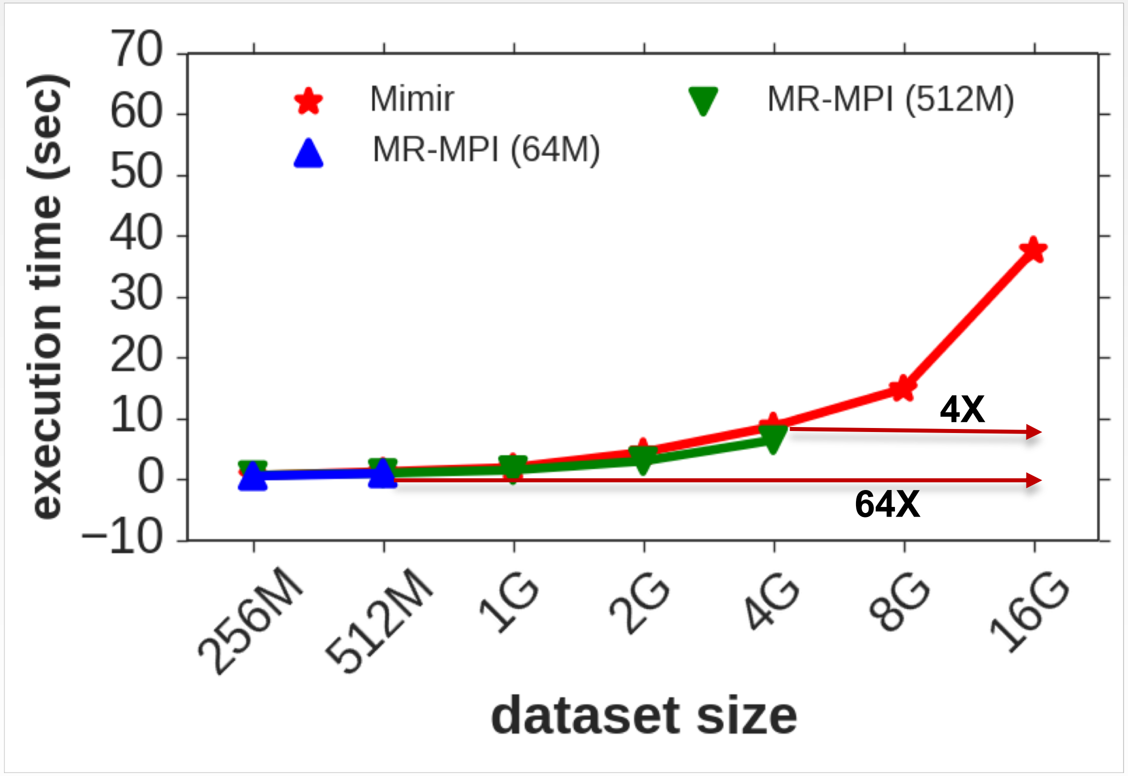
- **MIMIR**
  - Mimir can handle 16 X larger dataset in-memory compared with MR-MPI
  - Mimir scale to 16,384 processes
  - Mimir is a open-source *https://github.com/TauferLab/Mimir.git*

- *[1] T. Gao, Y. Guo, B. Zhang, P. Cicotti, Y. Lu, P. Balaji, and M. Taufer. Mimir: Memory-Efficient and Scalable MapReduce for Large Supercomputing Systems. In Proceedings of the IPDPS, 2017.*

Mimir vs. MR-MPI: WordCount on Comet

- Single-node execution (24 processes, 128G memory)
  - Benchmarks: WC with Wikipedia dataset
  - Settings: MR-MPI (64M page and 512M page); Mimir (64M page)

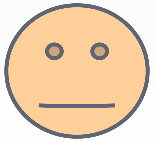
Mimir can handle **4X larger** dataset



But, can I run my Spark program on it…?

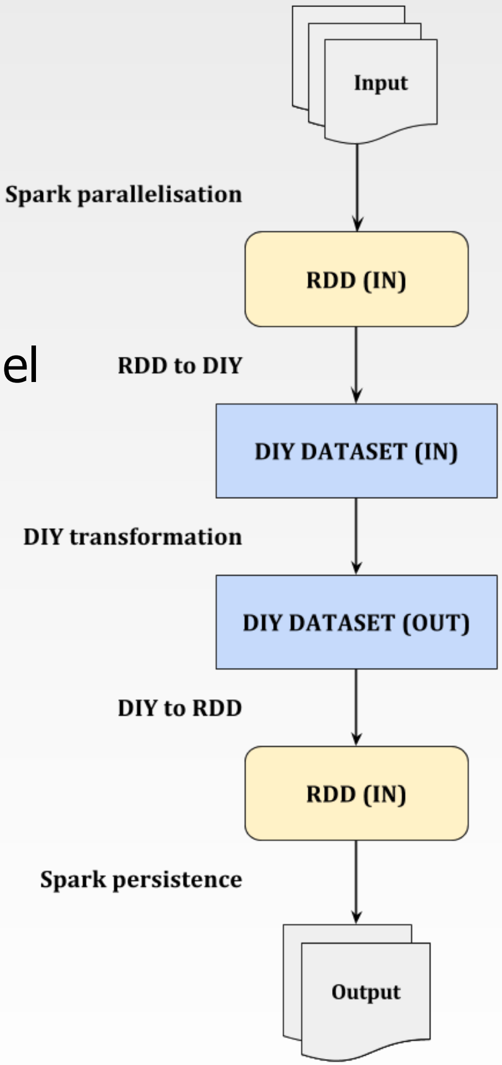
- The answer is no…
  - Programming environments do not match.
- Uuppps! The users are not very happy.

- What can we do?
  - Program in Spark and transparently jump to MPI world
  - How: using the RDD anstraction of Spark and the topologies of MPI
  - Is there a solution? Not yet, working on it: ARCOS + Argonne Labs
  - A fist proof of concept is running. Happy ? Not yet, but …

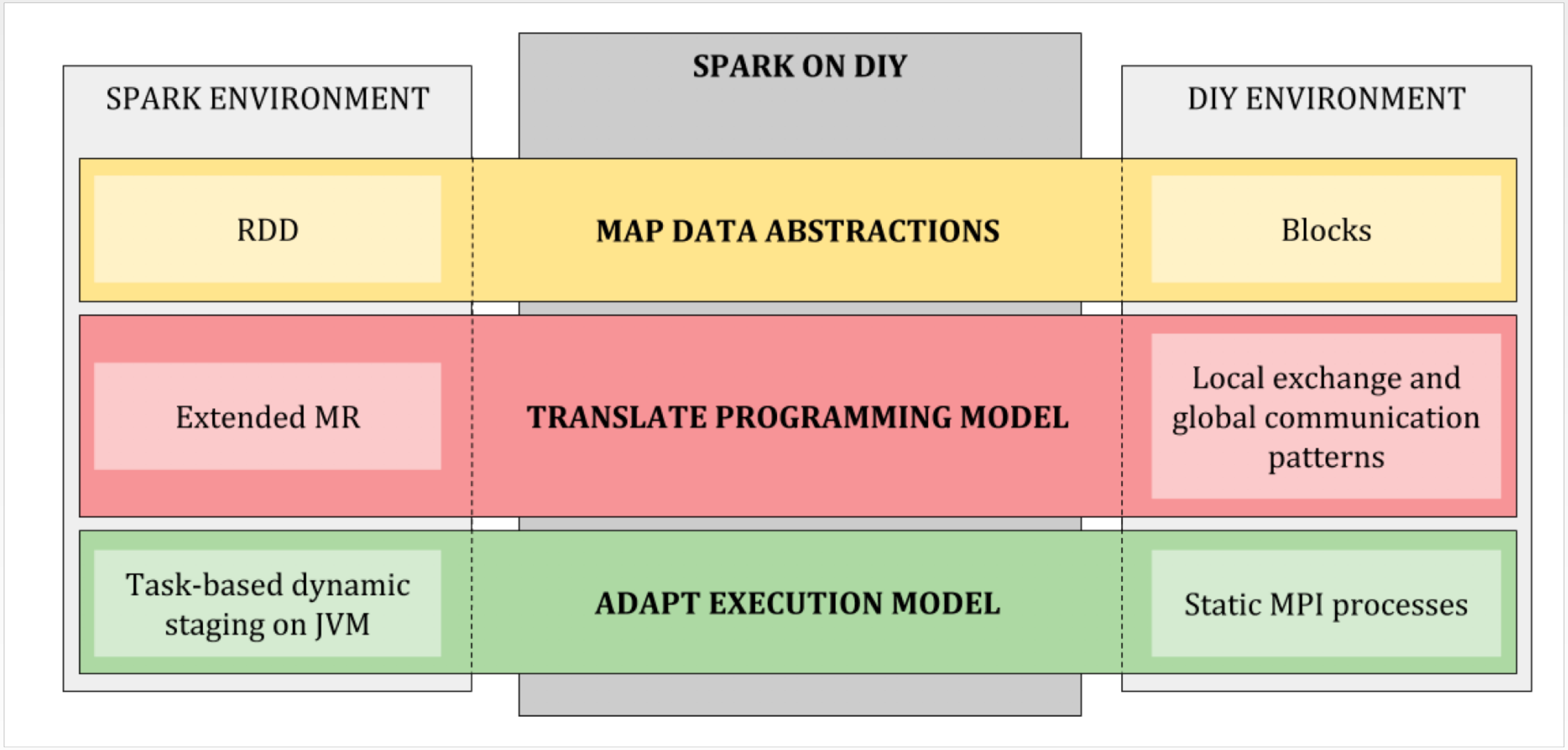


Proposal: connecting MPI and Spark

- To layer the Spark application model on top of a MPI-based library
- Goals:
  - Minimise the user knowledge of the underlying data model
  - Expose explicit interoperability
  - Preserve the nature of the Spark interface
  - Support multiple data types
- Platform:
  - Minimise data transfers
  - Integrate with the framework without changes



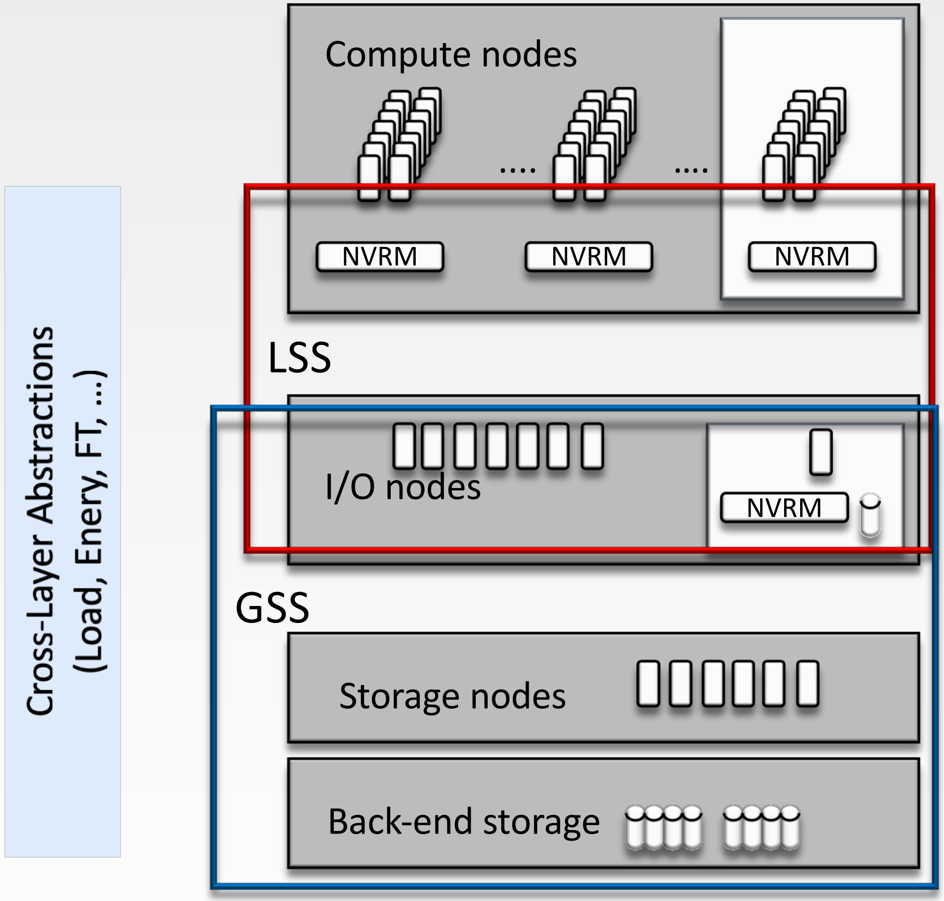
Spark on MPI


SPARK ENVIRONMENT
SPARK ON DIY
DIY ENVIRONMENT
RDD
MAP DATA ABSTRACTIONS
Blocks
Extended MR
TRANSLATE PROGRAMMING MODEL
Local exchange and global communication patterns
Task-based dynamic staging on JVM
ADAPT EXECUTION MODEL
Static MPI processes

Convergence in storage system is also needed

- Roles of storage in HPC systems
  - Data collection I/O
  - Analysis I/O. Logging.
  - Defensive I/O. Checkpointing
- Big Data requires:
  - Near-storage
  - Replication and
  - Elasticity

Scalable I/O system architecture



- Hybrid RAM/NVRAM local storage
- Active participation in data and metadata management
- Use many-core nodes computational power and fast inteconnection network
- Scalability with system for some workloads (burst scheduling)

- Hybrid NVRAM/HD
- Burst buffers for absorbing peaks of load
- Intermediate storage for (small) temporal loads

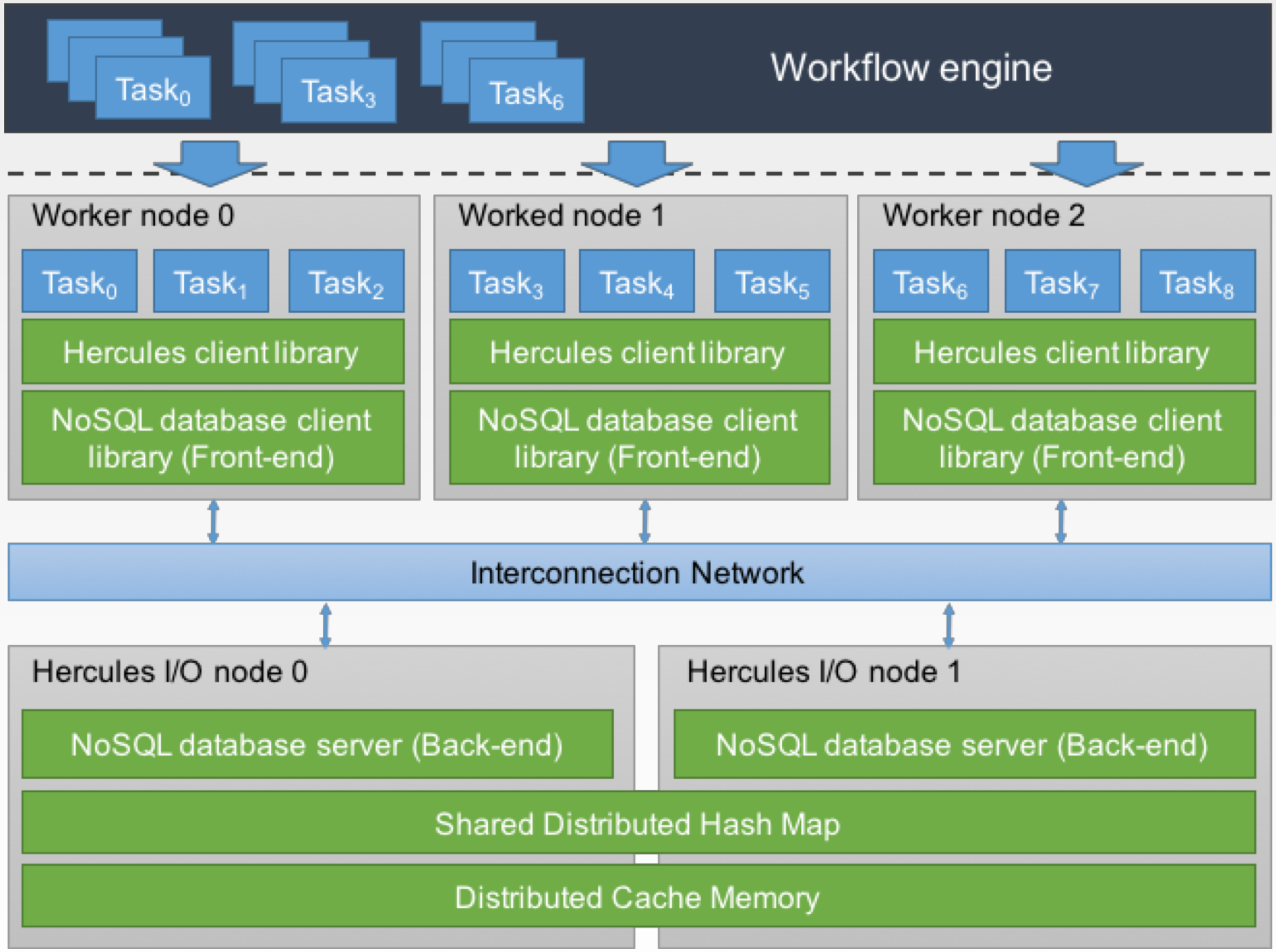
- Parallel/distributed file system (e.g.: GPFS, Lustre, PVFS)
- Global system image
- Storage object devices
- High performance storage access

Ad-hoc (in-memory) local storage

- Dynamic deployment of I/O tools needed
  - Application guided, less metadata
  - Mostly memory based (but finally persistent)
  - Static hierarchical FS will not do it (alone)

- Need to enhance data locality with load-balance in application execution
  - Computing and data intensive computing on same systems (HPC, HTC and workflows)
  - Process in-site, don´t store temporal data (to GSS)

LSS proposal: Hercules



Data-aware scheduling

- Now:
  - HDFS and algorithmic hashing placemente in Big Data
  - Optimization of load balance in HPC
- We need to be data locality-aware
  - Place RDD in node memory or local storage.
  - Execute MPI/analytic tasks in the node containing the data
- Problem: to know where data is to keep load balance:
  - Data-aware placement
    - Connect scheduler with Hercules to ask for data allocation

Problem: coordination LSS - GSS

- Vertical coordination
  - Map application models on storage models
  - Coordinate multiple level buffering/caching for latency hiding
  - Vertical data flow control: compute nodes <-> I/O nodes <-> file system <-> storage
    - Multiple-level write-back / write-though, Multiple-level prefetching
- Horizontal coordination
  - Collective I/O on compute nodes
  - I/O storage, aggregation, and operations on the I/O nodes

F. Isaila et all. Design and evaluation of multiple level data staging for Blue Gene systems. In IEEE Trans. of Parallel and Distributed Systems, 2011.

Conclusions

- Upcoming HPC and Big Data applications need hybrid infrastructures and execution platforms.
- Storage platforms convergence among HPC and BD is a must
  - Integrate with memory-centric ad-hoc storage systems.
  - Create mechanisms to induce data locality in HPC-oriented paradigms.
- Co-location of data and computation to improve performance
  - HPC scheduler must be data locality-aware
  - Data allocation must be CPU-aware
- We need efficient collective communication mechanisms in Big Data platforms
  - Joining MPI ang Spark models through RDDs

Universidad Complutense de Madrid

Towards Unification of HPC and Big Data Paradigms

Jesús Carretero

Computer Science and Engineering Department
University Carlos III of Madrid
jcarrete@inf.uc3m.es