Parallel Computing

Software

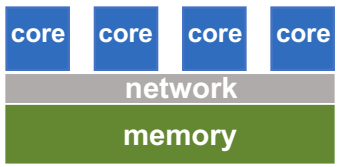
Introduction

1. Parallel Software
    a. Hardware-Native Programming Concepts
    b. Native Software and Interfaces
    c. R Interfaces to Low-Level Native Tools

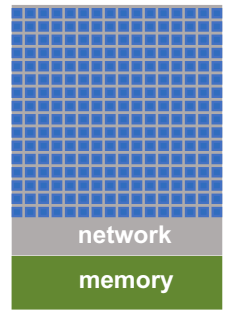
2. Programming Paradigms
    a. HPC and ABDS Software Stacks
    b.
    c.

Three Flavors of Parallel Hardware

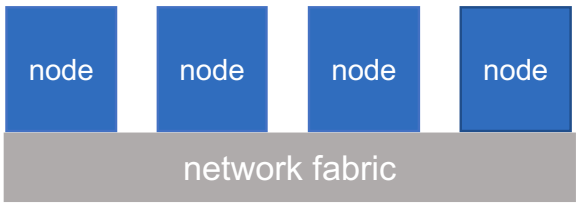
Shared Memory
Multicore Processor



Shared Memory
Co-Processor



GPU

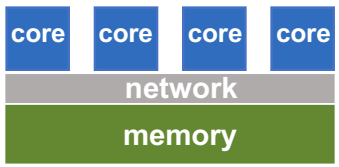


Distributed Memory Cluster

Native Programming Paradigms

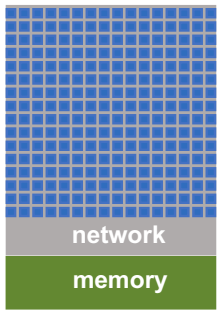
Default is serial: which tasks should be made parallel?

Shared Memory
Multicore Processor

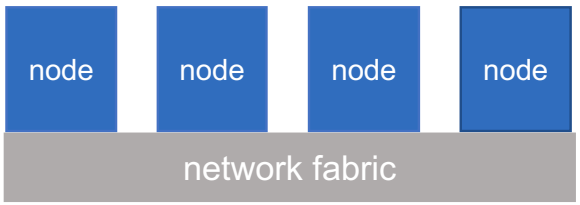


Offload data and tasks: We are slow but many!

Shared Memory
Co-Processor



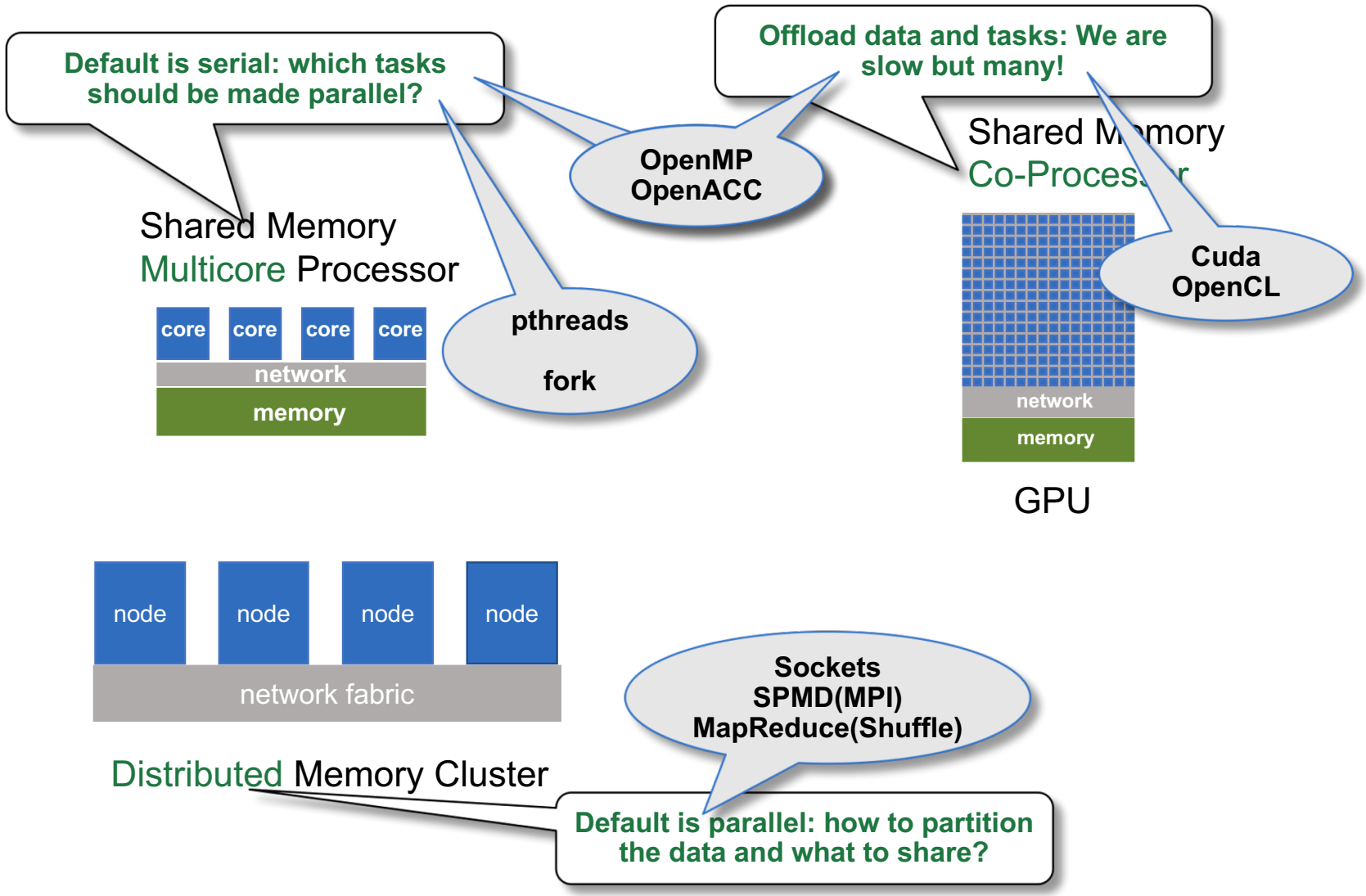
GPU



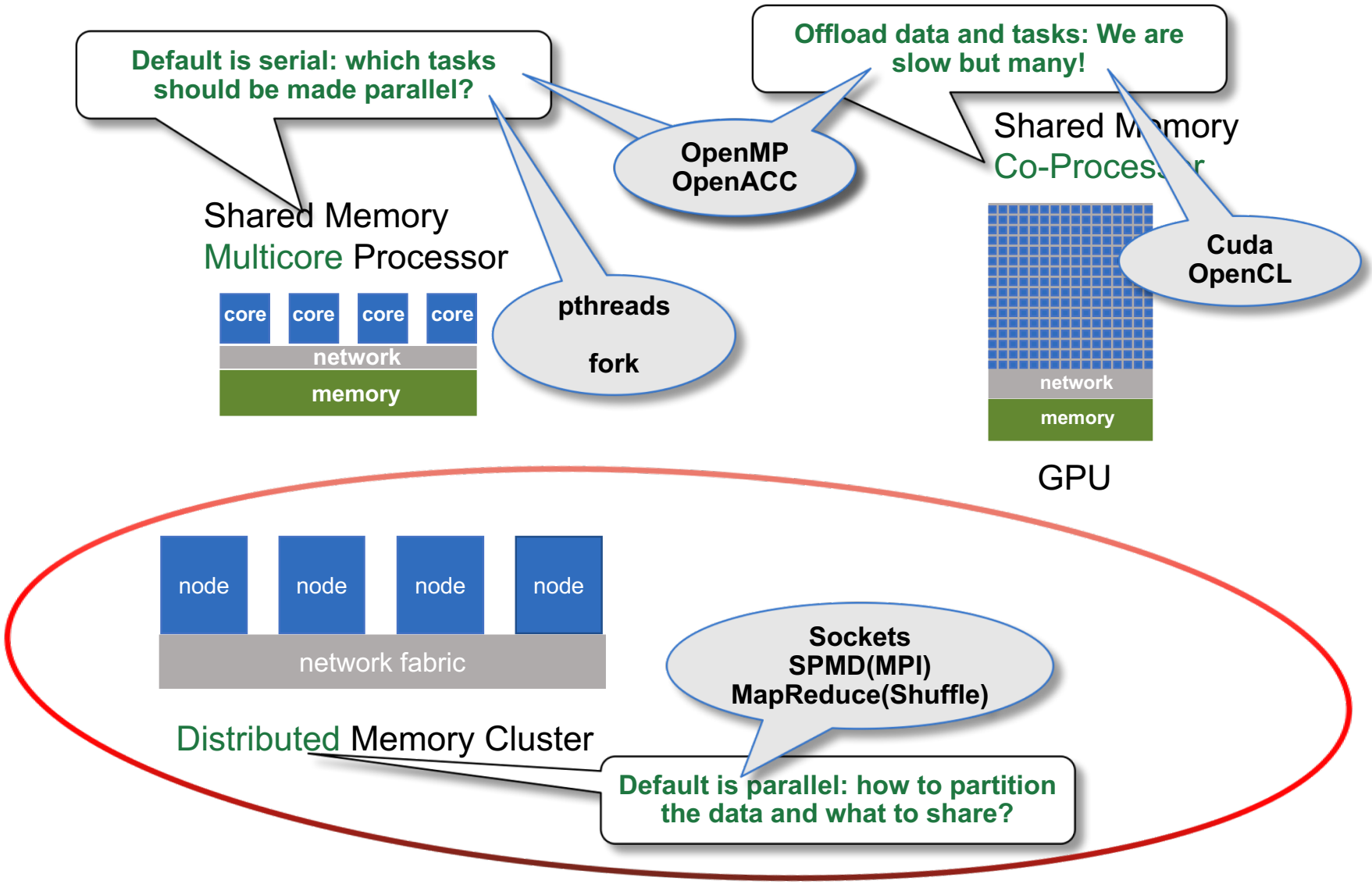
Distributed Memory Cluster

Default is parallel: how to partition the data and what to share?

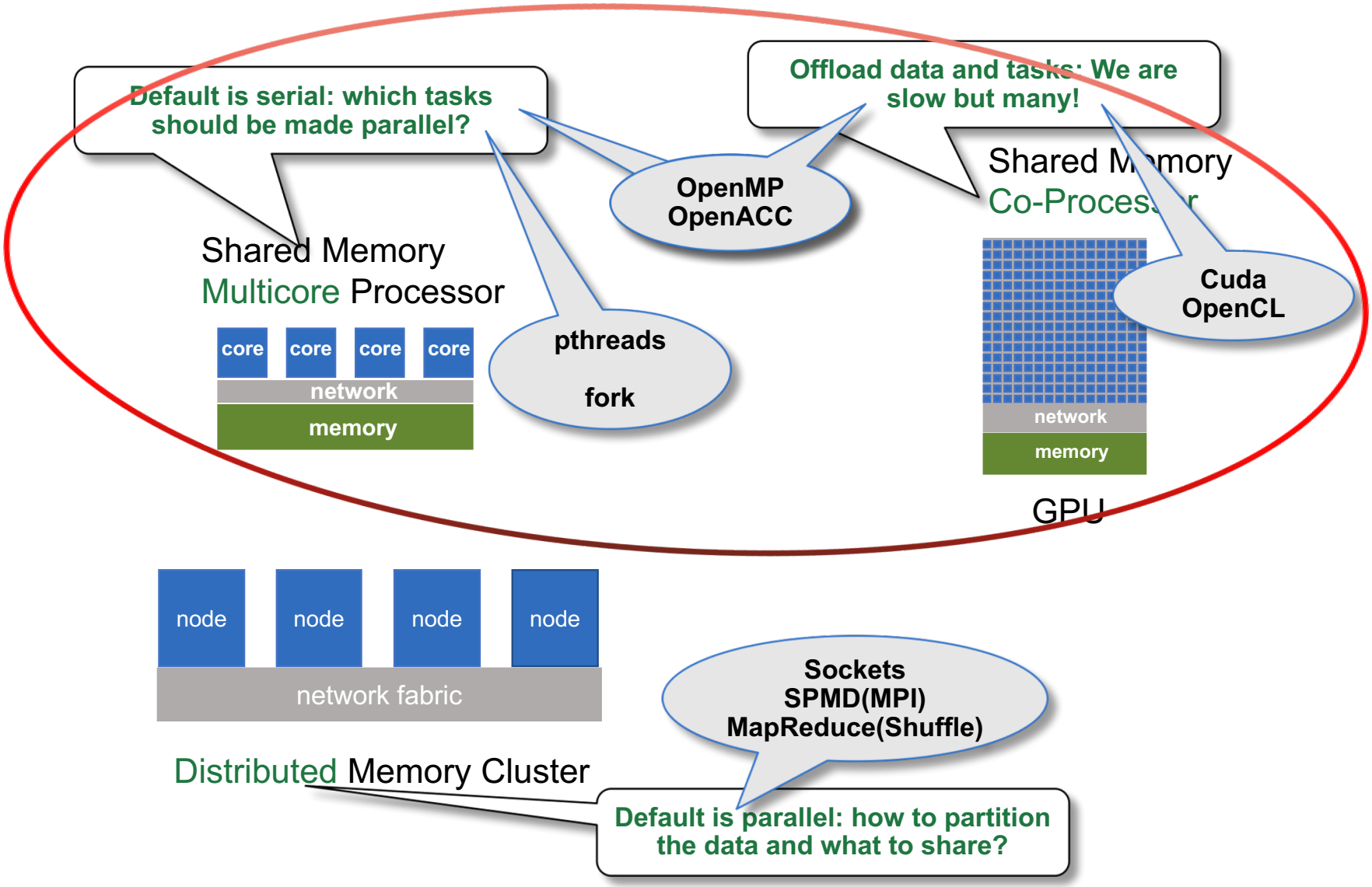
Native Software and Interfaces



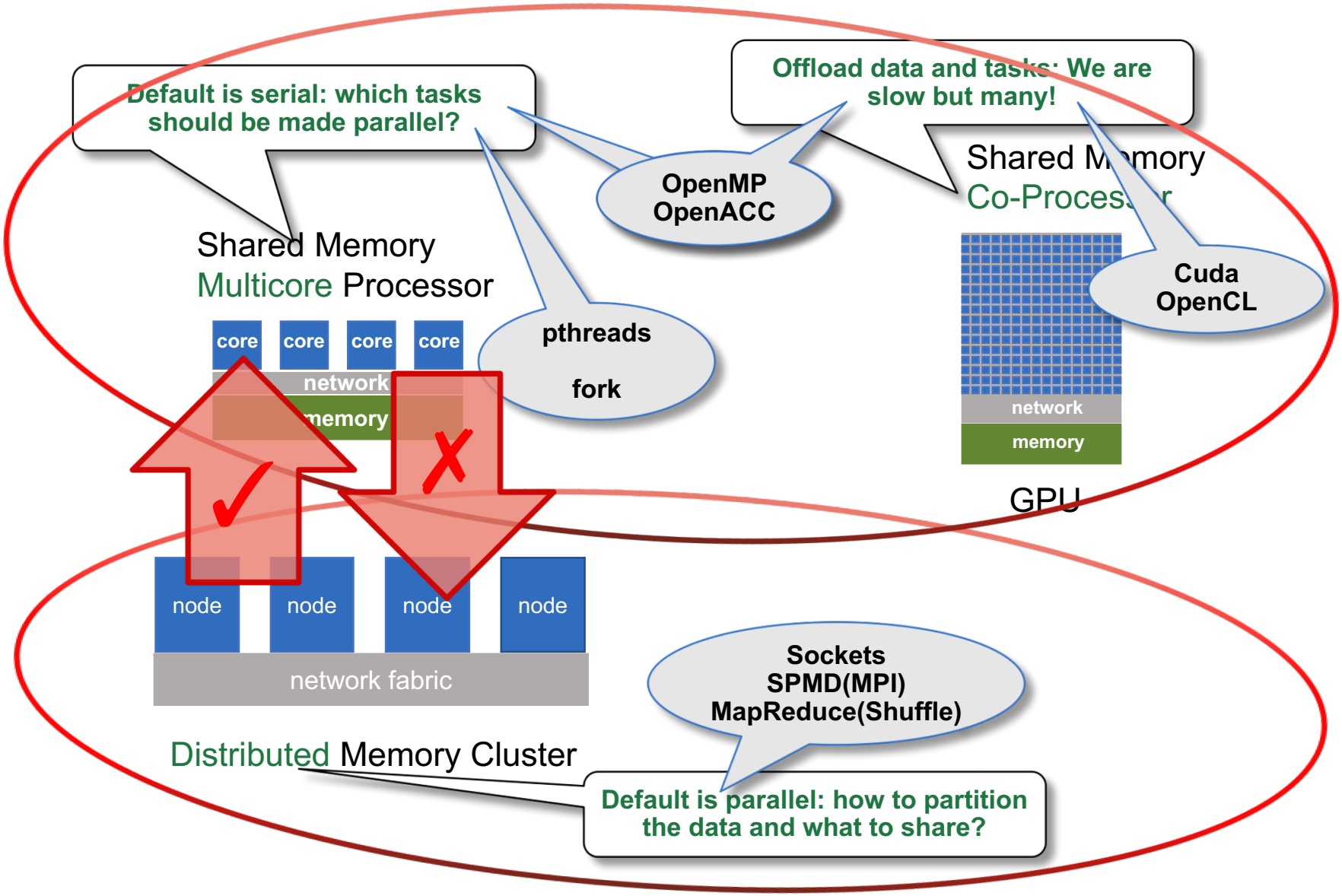
30+ Years of Parallel Computing Research



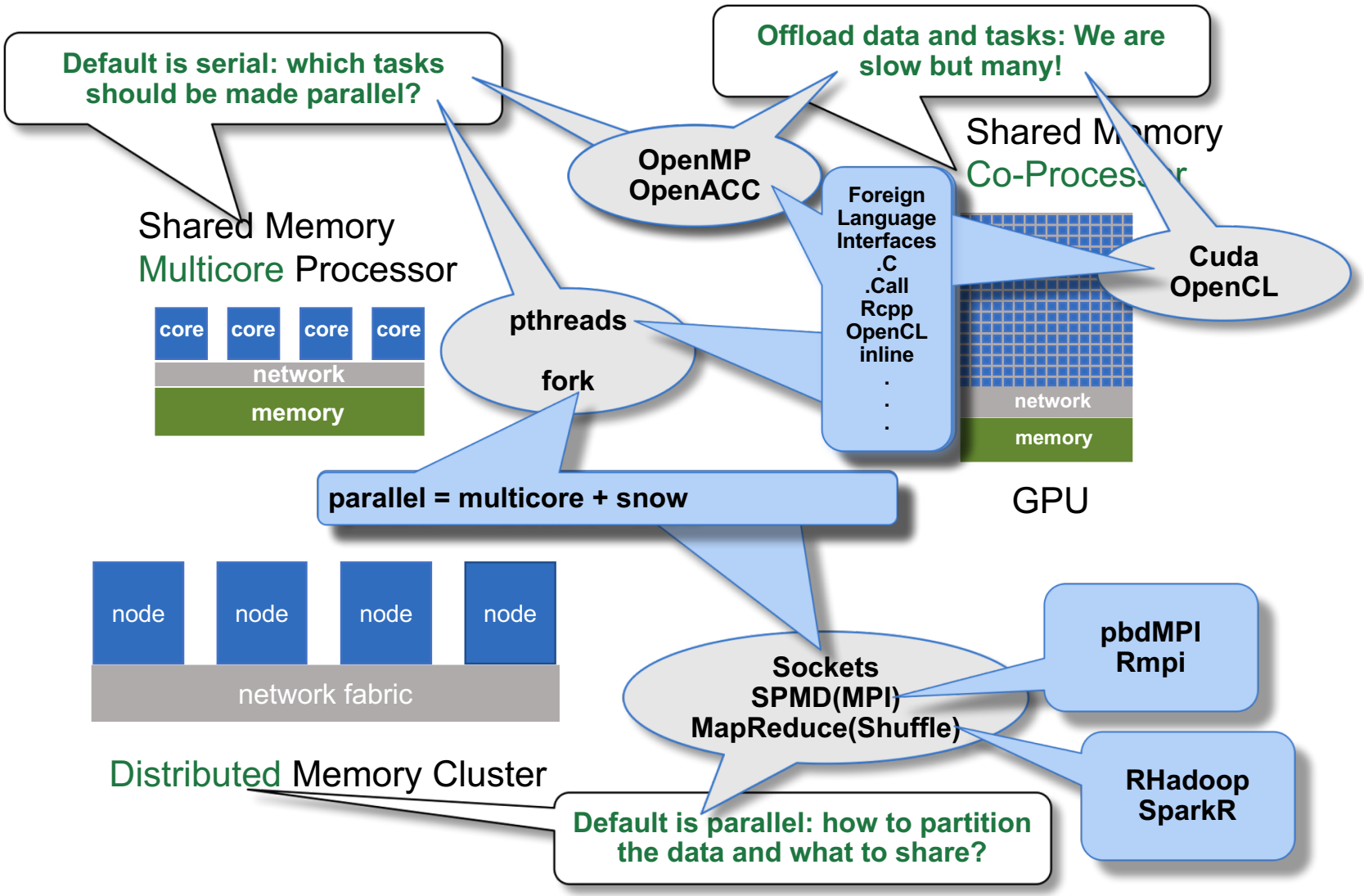
Last 12 Years of Advances



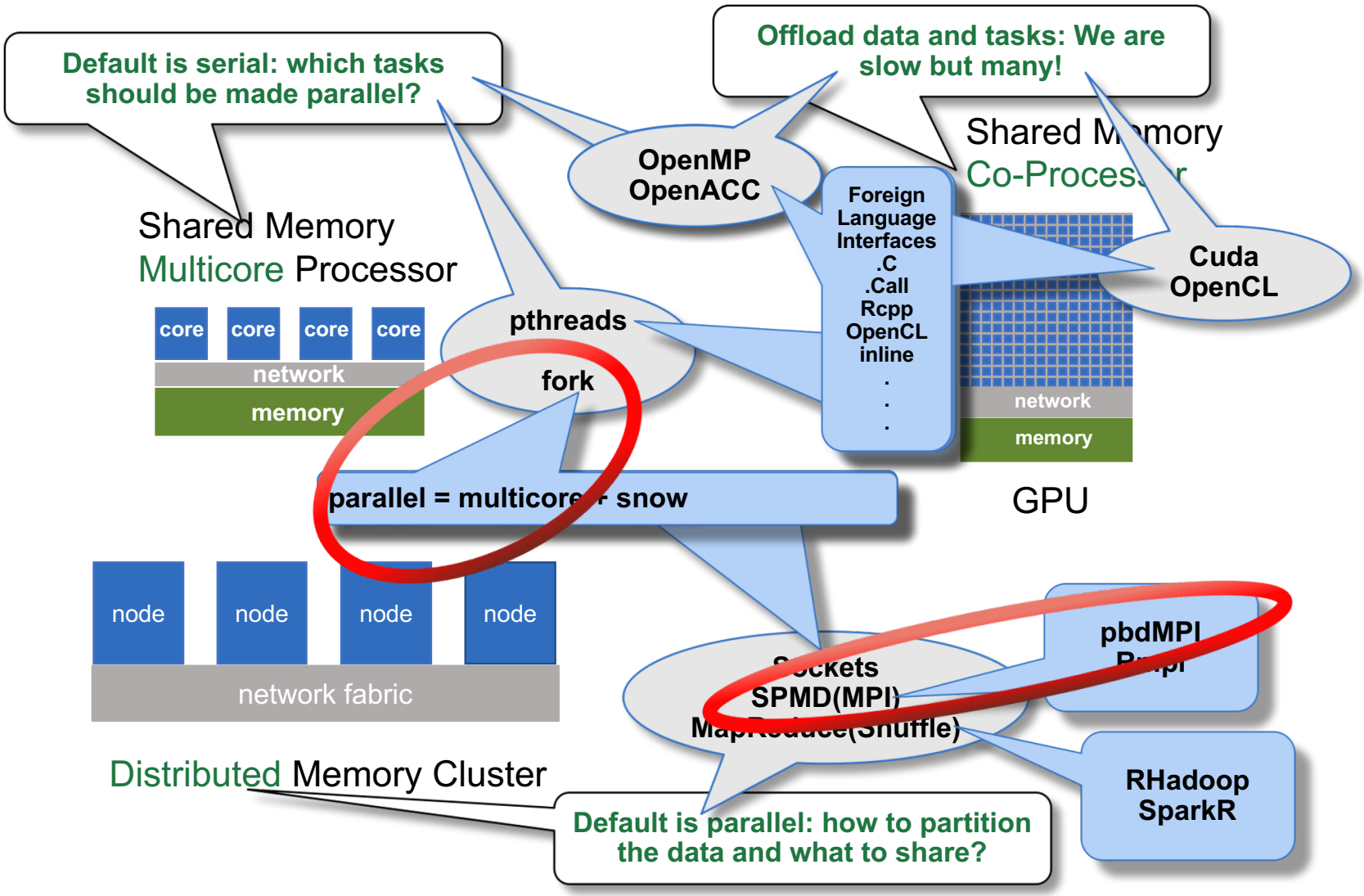
Distributed Programming Works in Shared Memory



R Interfaces to Low-Level Native Tools



We Cover: multicore(parallel) and pbdMPI



Manager-Workers

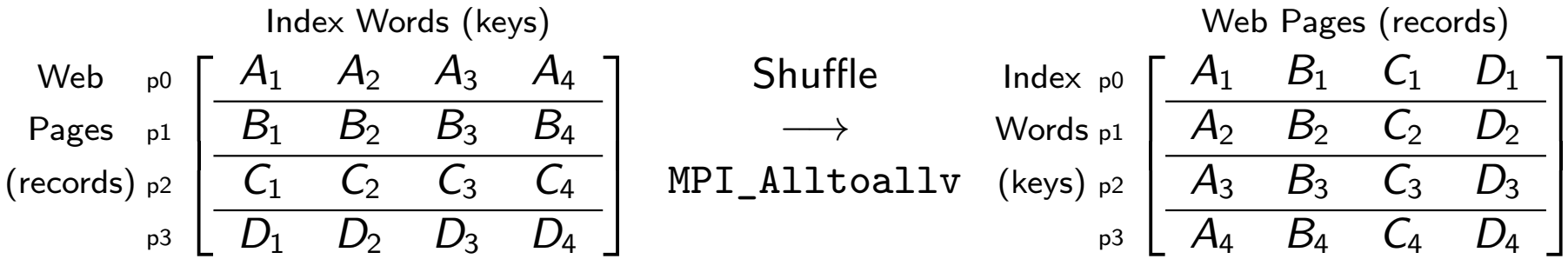
- Easy paradigm that is native to multicore
  - A serial program (Manager) divides up work and/or data
  - Manager sends work (and data) to workers
  - Workers run in parallel without interaction
  - Manager collects/combines results from workers
- Aligns with natural (serial) thought process
  - Default is serial, ask for parallel
  - Manager is serial, workers are parallel
  - Divide-Recombine fits this model
  - Single interactive user
  - Client-server

MapReduce

- A batch processing concept
- Born of a web search engine
- Decouples some coupled problems with communication: shuffle
- User decomposes computation into parallel Map and Reduce steps
- Writes two serial codes: Map and Reduce

MapReduce: a Parallel Search Engine Concept

Search MANY documents **Serve MANY users**



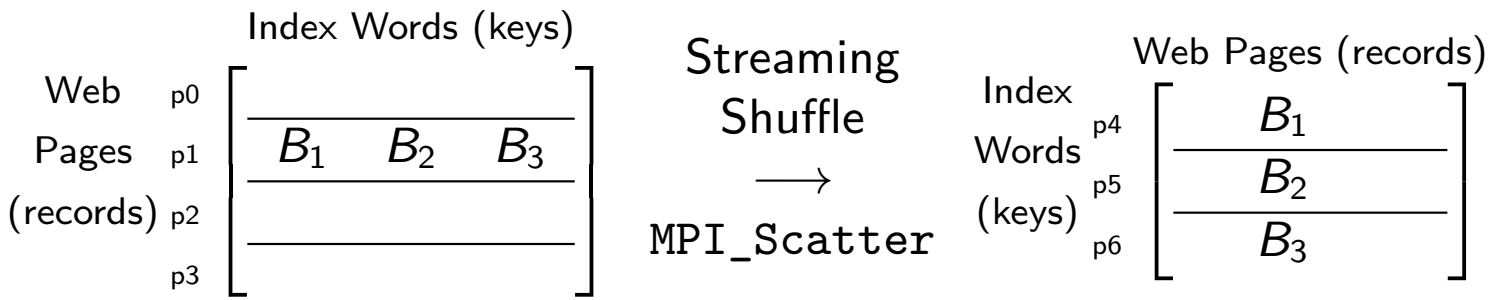
Shuffle is a block matrix transpose?

More documents = more processors

More users = more processors

Size flexibility in shuffle

Dynamically add or remove processors based on demand



SPMD: Single Program Multiple Data

Write one general program so many copies of it can run asynchronously and cooperate (usually via MPI) to solve the problem.

- The prevalent way of distributed programming in HPC for 30+ years
- Can handle tightly coupled parallel computations
- It is designed for batch computing
- There is usually no manager - rather, all cooperate
- Prime driver behind MPI specification
- Way to program server side in client-server

$$A = X^T X = \sum_i X_i^T X_i \text{, where } X = \begin{bmatrix} X_1 \\ \vdots \\ X_8 \end{bmatrix} \qquad \textbf{(Blocks of rows)}$$

SPMD and recursive doubling predate MPI

1986 hypercube: Reduction by looping over bitflips and dimensions

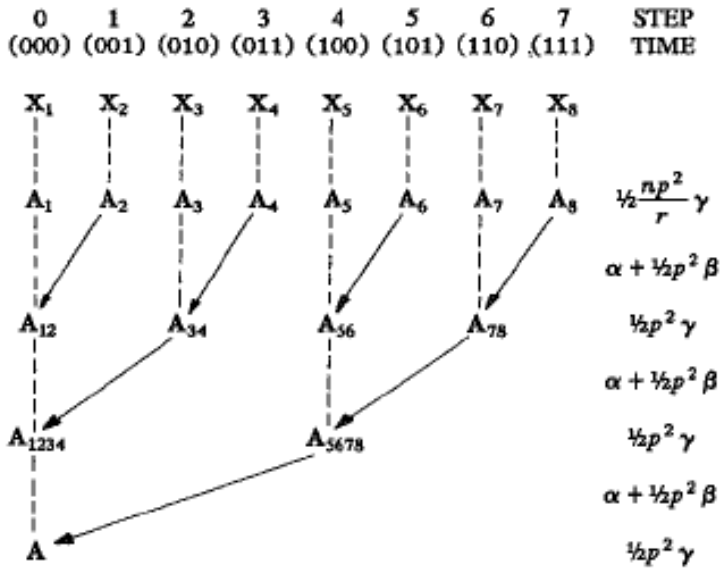


FIG. 4. *Computation of* $\mathbf{A} = \mathbf{X}'\mathbf{X}$ *on an 8-processor hypercube, with final result on processor 0.*

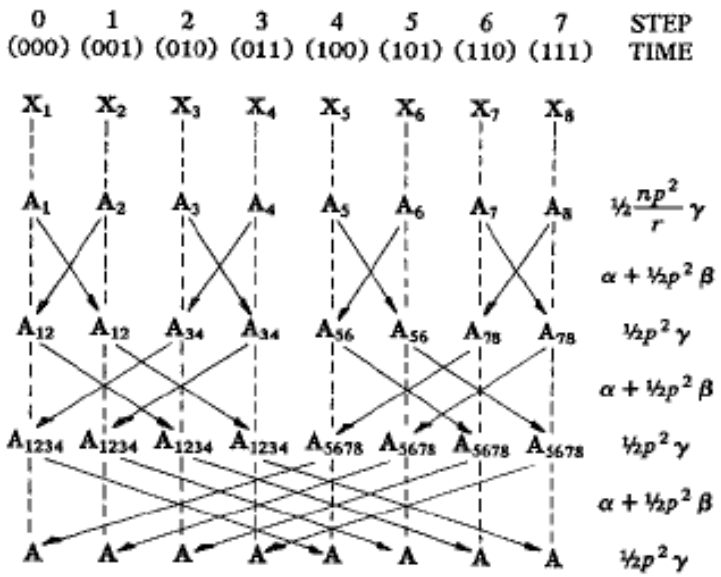


FIG. 6. *Computation of* $\mathbf{A} = \mathbf{X}'\mathbf{X}$ *on an 8-processor hypercube, with final result on all processors.*

1

[1] Ostrouchov (1987). Parallel Computing on a Hypercube: An overview of the architecture and some applications. *Proceedings of the 19th Symposium on the Interface of Computer Science and Statistics*, p.27-32.

$A = X^T X = \sum_i X_i^T X_i$, **where** $X = \begin{bmatrix} X_1 \\ \vdots \\ X_8 \end{bmatrix}$ **(Blocks of rows)**

MPI and pbdMPI bring simplicity to SPMD

Today: A = reduce(crossprod(X)) A = allreduce(crossprod(X))

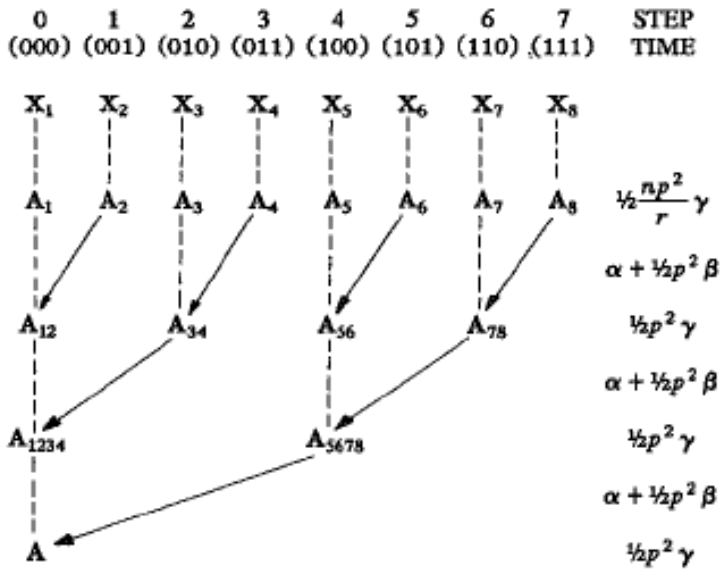


FIG. 4. *Computation of* $\mathbf{A} = \mathbf{X}'\mathbf{X}$ *on an 8-processor hypercube, with final result on processor 0.*

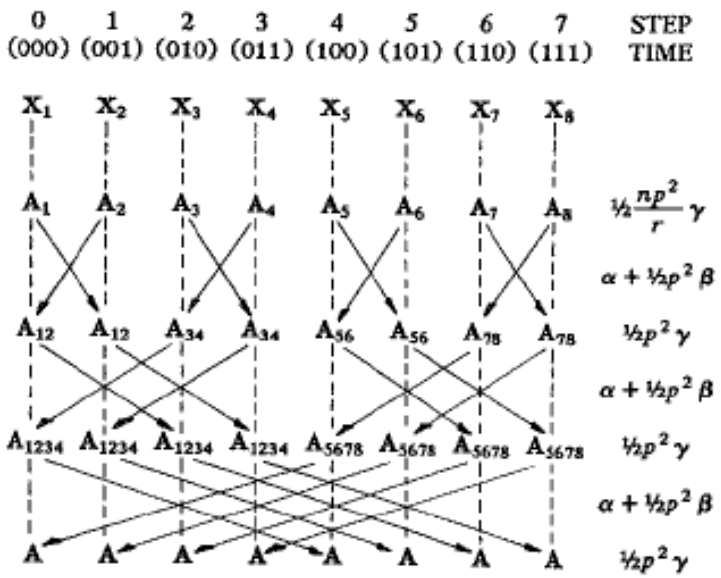


FIG. 6. *Computation of* $\mathbf{A} = \mathbf{X}'\mathbf{X}$ *on an 8-processor hypercube, with final result on all processors.*

2

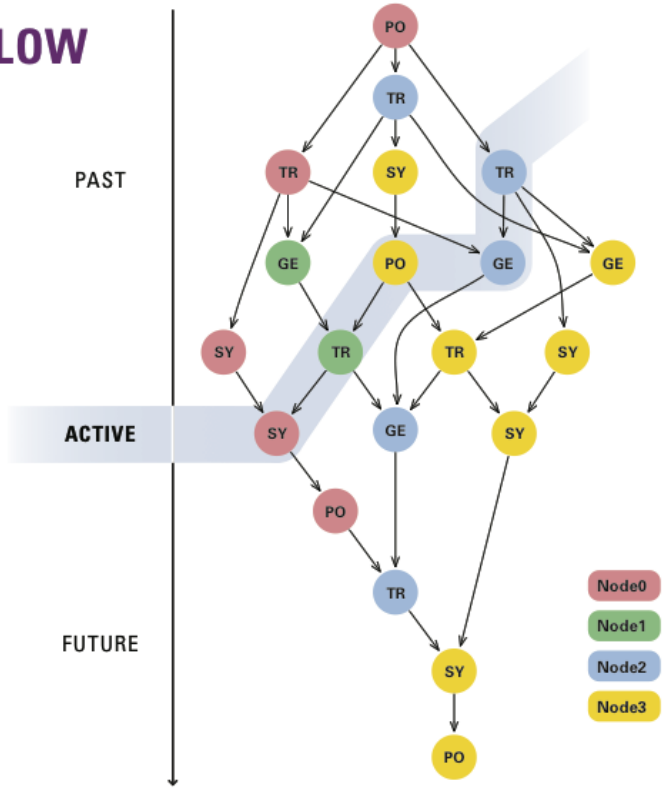
[2] Ostrouchov (1987). Parallel Computing on a Hypercube: An overview of the architecture and some applications. *Proceedings of the 19th Symposium on the Interface of Computer Science and Statistics*, p.27-32.

SPMD and MapReduce (MPI and Shuffle)

- One parallel code or pairs of serial codes
  - SPMD: write one code that generalizes the serial code
  - MapReduce: decompose serial code into pairs of map and reduce serial functions and a control code
- Concepts differ in approach to communication
  - SPMD makes communication explicit, gives choices (MPI)
  - MapReduce hides communication, uses one choice (shuffle)
  - MPI reduce operations take advantage of recursive doubling:
    - Most reductions are associative and commutative
    - Reducing n items takes $\log_2(n)$ steps
    - Vendor MPI implementations are architecture-aware
    - Matrix computations orders of magnitude faster

Data-flow: Parallel Runtime Scheduling and Execution Controller (PaRSEC)

EFFICIENT DATA FLOW REPRESENTATION



Graphic from icl.cs.utk.edu

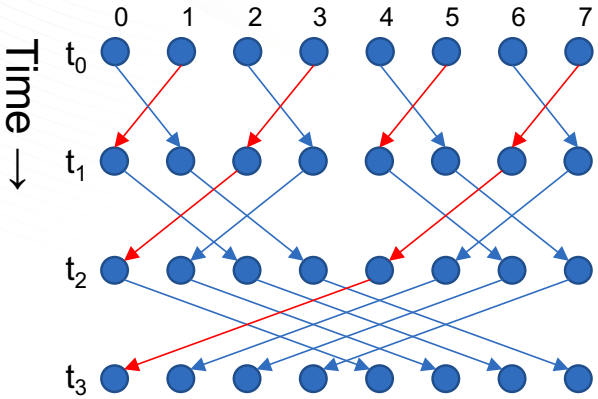
FEATURES
Supports Distributed Heterogeneous Platforms
Sustained Performance
NUMA & Cache Aware Scheduling
State-of-the-art Algorithms
Capacity Level Scalability
Performance Portability
Implicit Communication
Communication Overlapping

Bosilca, G., Bouteiller, A., Danalis, A., Faverge, M., Herault, T., Dongarra, J. "PaRSEC: Exploiting Heterogeneity to Enhance Scalability," IEEE Computing in Science and Engineering, Vol. 15, No. 6, 36-45, November, 2013.

- Master data-flow controller runs distributed on all nodes
- Dynamic generation of current level in flow graph
- Effectively removes collective synchronizations for additional speedup

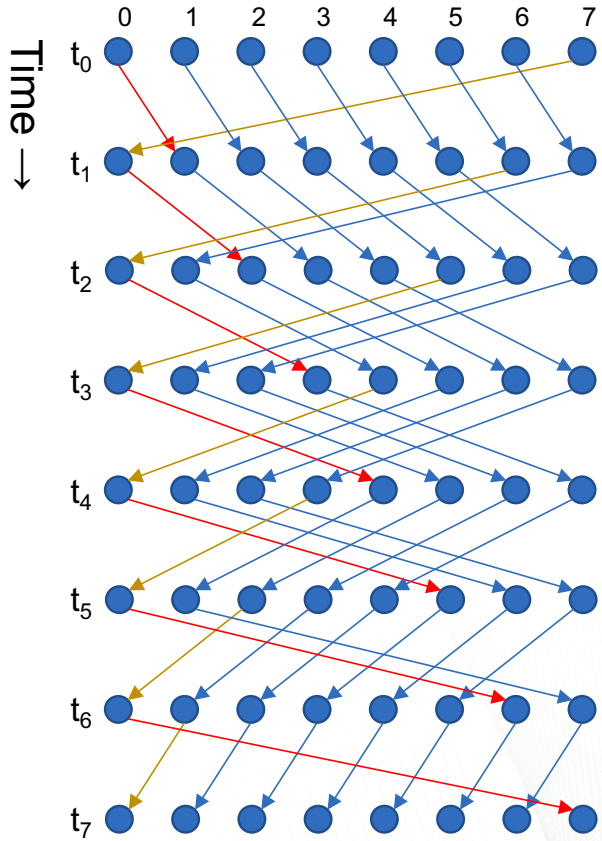
Recursive Doubling Advantage of MPI

allreduce, reduce, allgather, gather



Communication steps when reduction is associative		
Nodes	shuffle	allreduce
8	**14**	**3**
128	**254**	**8**
n	**2*(n – 1)**	**$\log_2 n$**

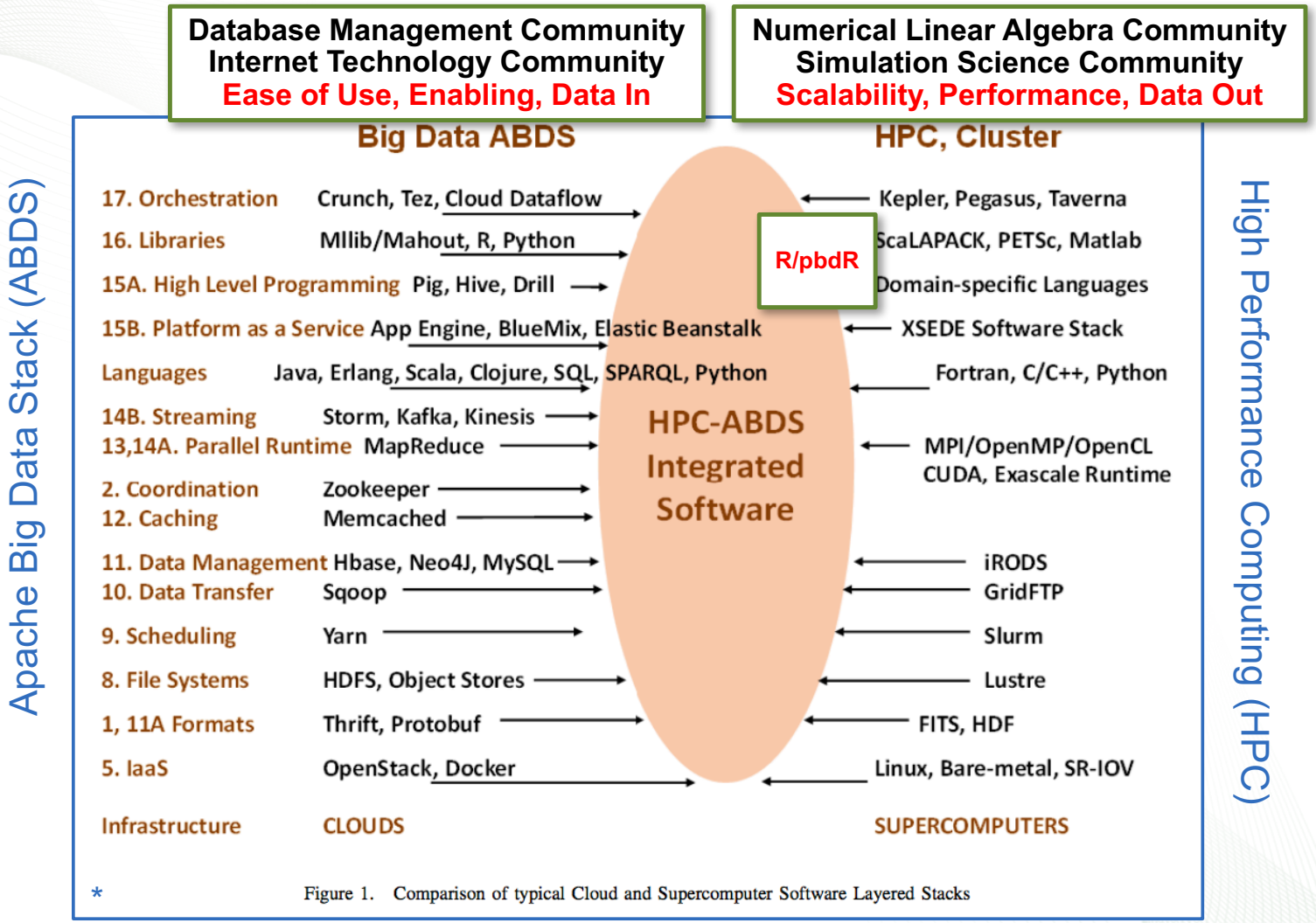
shuffle (gather-scatter)



OAK RIDGE National Laboratory

1

Convergence of Cloud and HPC for Data



* Figure 1. Comparison of typical Cloud and Supercomputer Software Layered Stacks

*Figure from: G. C. Fox, J. Qiu, S. Kamburugamuve, S. Jha and A. Luckow, "HPC-ABDS High Performance Computing Enhanced Apache Big Data Stack," *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Shenzhen, 2015, pp. 1057-1066. doi: 10.1109/CCGrid.2015.122

OAK RIDGE National Laboratory