膨大データ向けの新しい並列プログラミングフレームワーク

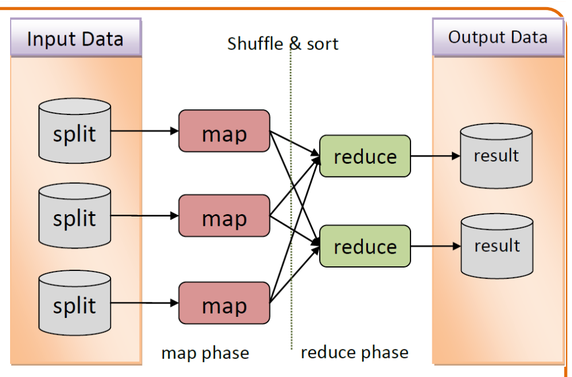
A New Parallel Programming Framework for Processing Large-scale Data

劉 雨　　胡 振江

Yu LIU　Zhenjiang HU

Background and Objective

MapReduce is a popular framework for data-intensive distributed computing. It uses a simple but efficient divide-and-conquer fashion to harnesses the power of large clusters of computers.



The **list homomorphisms** are a class of recursive functions on lists, which serve well with D&C paradigm and can be efficiently implemented in parallel.

$$\begin{aligned} h\ [a] &= f\ a \\ h\ (x +\!\!+\ y) &= h(x) \odot h(y) \end{aligned}$$

To resolve many computation problems that are difficult to be programmed by *MapReduce*, we propose **a homomorphism-based framework** to provide a systematical solution of automatically generating efficient *MapReduce* programs.

The 3rd homomorphism theorem

$$\begin{aligned} h\ [a] &= f\ a \\ h\ (x +\!\!+\ y) &= h(x) \odot h(y) \end{aligned}$$

iff

$$\begin{aligned} h\ [a] &= f\ a \\ h\ ([a] +\!\!+\ x) &= a \oplus h\ x \\ h\ (x +\!\!+\ [a]) &= h\ x \otimes a. \end{aligned}$$

Our framework is base on the 3rd homomorphism theorem. By the 3rd homomorphism theorem, a list homomorphism can be got from two sequential functions.

Homomorphism-based Framework

By our framework, list homeomorphisms can be derived from user-input sequential functions, and automatically mapped to efficient *MapReduce* programs.

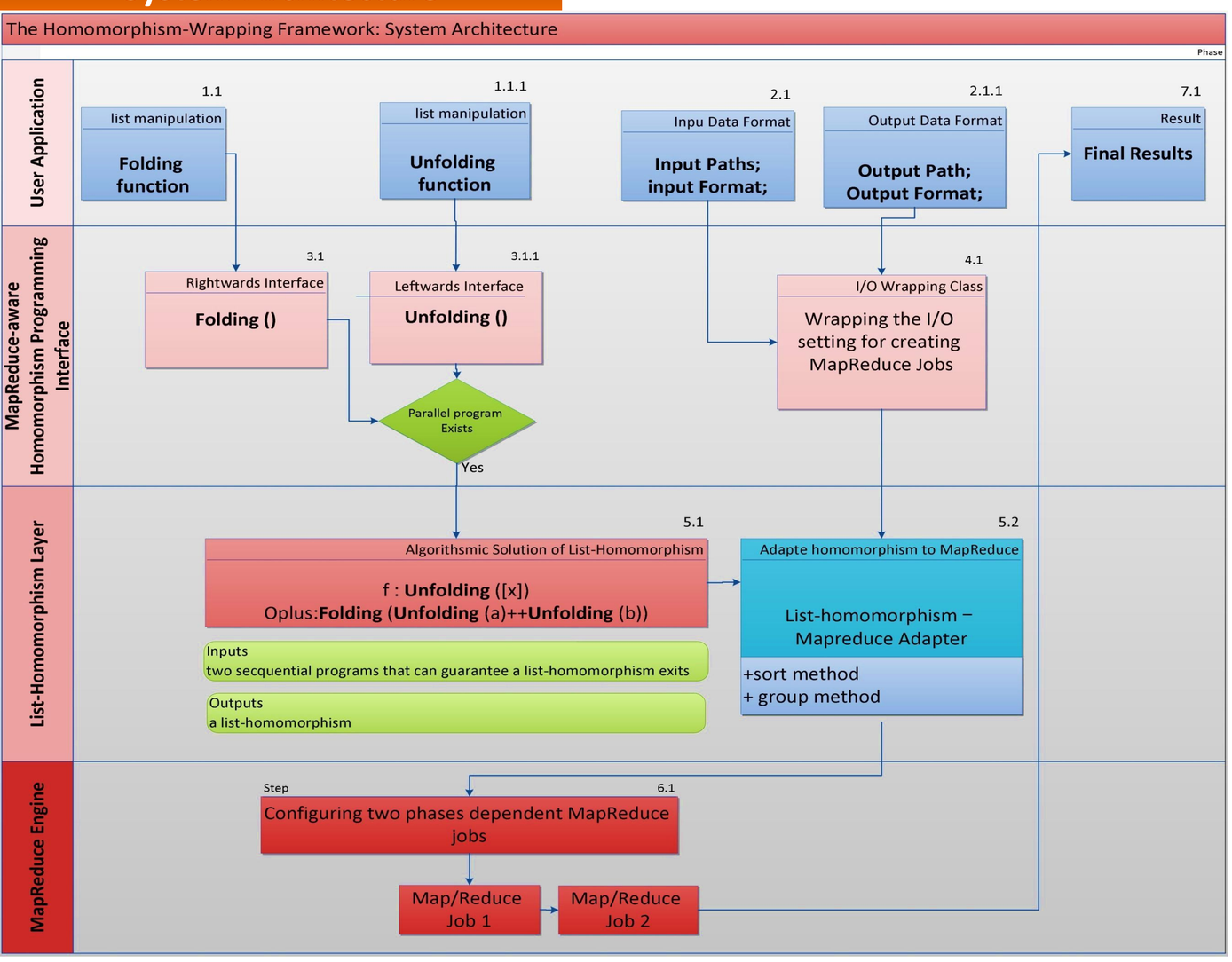
The Schematic Diagram

Programming interfaces
The user implements 2 sequential functions using system APIs.

→ *Internal derivation*
The corresponding list homomorphism is derived by framework.

→ *MapReduce implementation*
A chain of MapReduce jobs is generated, which implements the list homomorphism.

System Architecture



An Example

The maximum prefix sum problem

Compute the maximum of all the prefix sums, e.g.,

mps [1, 2, -1, 4, 3, -9] = 9

Folding function: h

Input: xs → h → return: (mps xs, sum xs)

Unfolding function: h°

Input: (m, s) → h° → return: [m , s-m]

➢ **Automatically Derived Homomorphism:**

A homomorphism *h* can be derived from above inputs:

$h = (\![f, \odot]\!)$ **where**

$$\begin{aligned} f\ a &= h\ [a] \\ l \odot r &= h\ (h^\circ\ l +\!\!+ h^\circ\ r) \end{aligned}$$

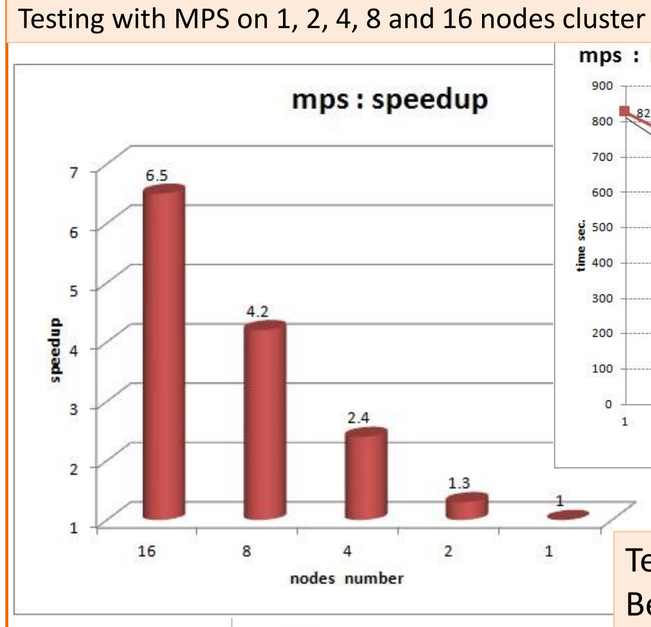
➢ **MapReduce implementation of list homomorphism**

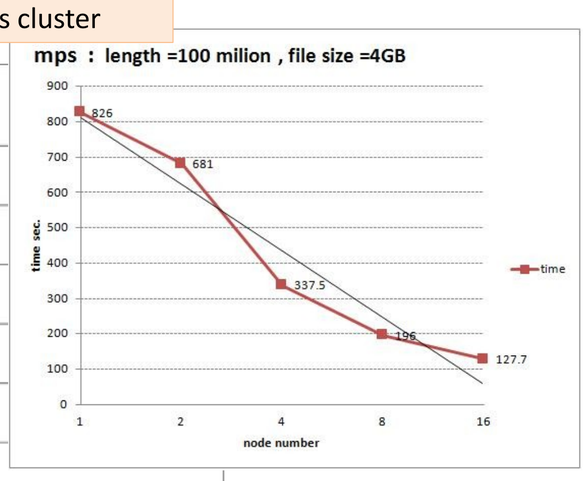
Through the algorithm that implements list homomorphisms using MapReduce, multi-phase map-reduce jobs are obtained and can be executed on Hadoop cluster.

Benchmark on Hadoop Cluster

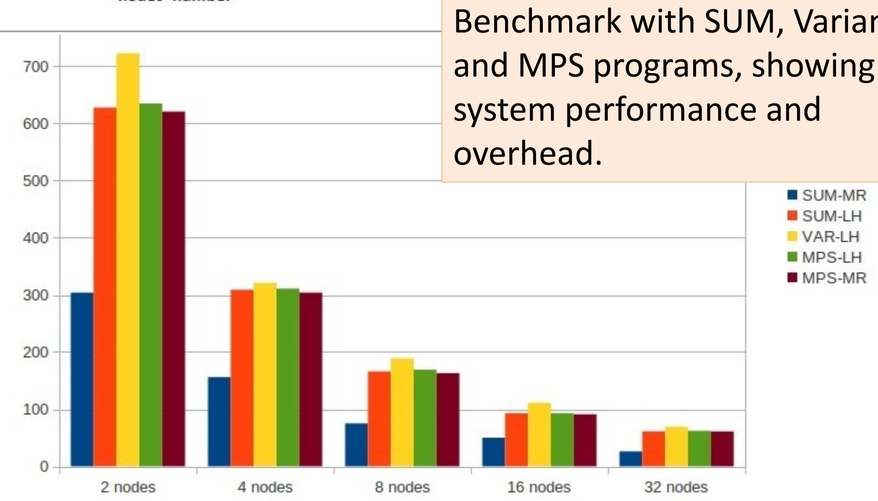
COE cluster of Tokyo University

Testing with MPS on 1, 2, 4, 8 and 16 nodes cluster





Tests on NII edubaseCloud Benchmark with SUM, Variance and MPS programs, showing the system performance and overhead.



Conclusions

- Efficiency: List homomorphism can be efficiently implemented with *MapReduce*.
- Programmability: The third homomorphism theorem makes it simple to develop parallel program with *MapReduce*.
- Practicability: The experiments of mps and others have shown the usefulness and power of list homomorphism framework.

Reference

Yu Liu, Zhenjiang Hu, Kiminori Matsuzaki, **Towards Systematic Parallel Programming over MapReduce** (Euro-Par'11).

NII

連絡先：　胡　振江（Zhenjiang HU）／ 国立情報学研究所 アーキテクチャ科学研究系　教授

TEL：03-4212-2530　　FAX：03-4212-2533　　Email：hu@nii.ac.jp