**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.

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

\[
\begin{align*}
 h[a] &= f(a) \\
 h(x + y) &= h(x) \odot h(y)
\end{align*}
\]

 iff

\[
\begin{align*}
 h[a] &= f(a) \\
 h(x + y) &= h(x) \odot h(y)
\end{align*}
\]

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 homomorphisms can be derived from user-input sequential functions, and automatically mapped to efficient MapReduce programs.

#### The Schematic Diagram

The user implements 2 sequential functions using system APIs.

The corresponding list homomorphism is derived by framework.

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

#### System Architecture

The homomorphism-Wrapping Framework: System Architecture

The maximum prefix sum problem

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

\[
\text{mps } [1, 2, -1, 4, 3, -9] = 9
\]

- **Automatically Derived Homomorphism:**
  A homomorphism \( h \) can be derived from above inputs:

\[
\begin{align*}
 f(a) &= h[a] \\
 l \odot r &= h(h^c l + h h^c r)
\end{align*}
\]

- **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).