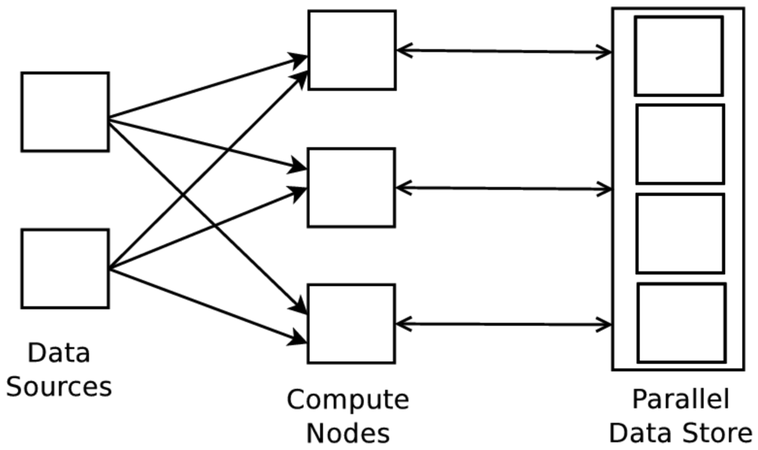
# Optimizing Remote Calls in Parallel Data Management Systems

**Bikash Chandra**
**Supervisor: Prof. S. Sudarshan**

## Architecture

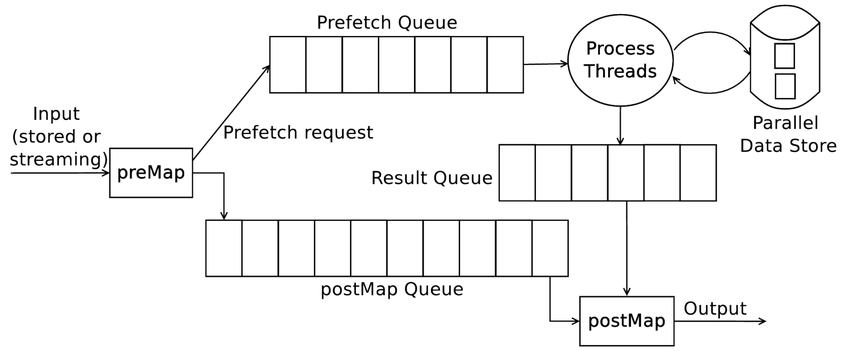



- Read data items from (streaming or stored) data sources
- Compute the function f(k,p), where
  - k- key for fetching values
  - p- list of parameters
- Fetch values from the parallel data store to compute the function

## Motivating Applications

- Entity annotation
  - Marking up text with the entities they refer to
  - Entities can be annotated on stored or streaming text
  - Models for classifications of entities are stored in data stores
- Genome read mapping
- … and many others

## Data access optimization




- Use techniques of batching and prefetching to optimize data access
- APIs to enable prefetch calls in a preMap thread for Hadoop and Muppet stream processing framework

## Optimizing Stored Procedure Invocation

**f(k,p)**

Frequent keys

Fetch and cache values from the data store.

- Function computation at compute node

**Online optimization**
Statistics are computed at runtime

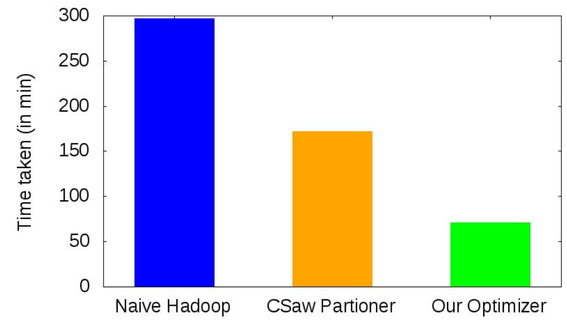
Non-frequent keys

Push computations to the data node

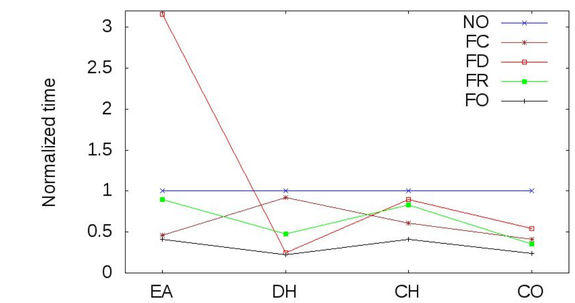
- Consider CPU and network costs and the load at the compute and data node
- Compute some functions at data node and return uncomputed values for remaining (in a load balanced way)

Mitigates skew due to data distribution and computational imbalance

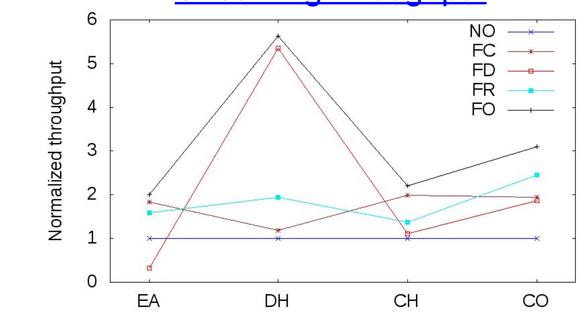
## Entity Annotation performance




## MapReduce performance




## Streaming throughput




NO- No optimization
FC- Function at compute node
FD- Function at data node
FR- Random comp/node
FO- Optimized

EA- Entity annotation
DH- Data Heavy
CH- Compute Heavy
CO- Combined data and compute heavy

Work supported by TCS PhD fellowship