ITask

Interruptable Tasks: Treating Memory Pressure As Interrupts for Highly Scalable Data-Parallel Programs

Lu Fang

Advisor: Guoqing(Harry) Xu

University of California, Irvine

UCIRVINE

Background and Motivation

Scalability issues remain common in Big Data systems

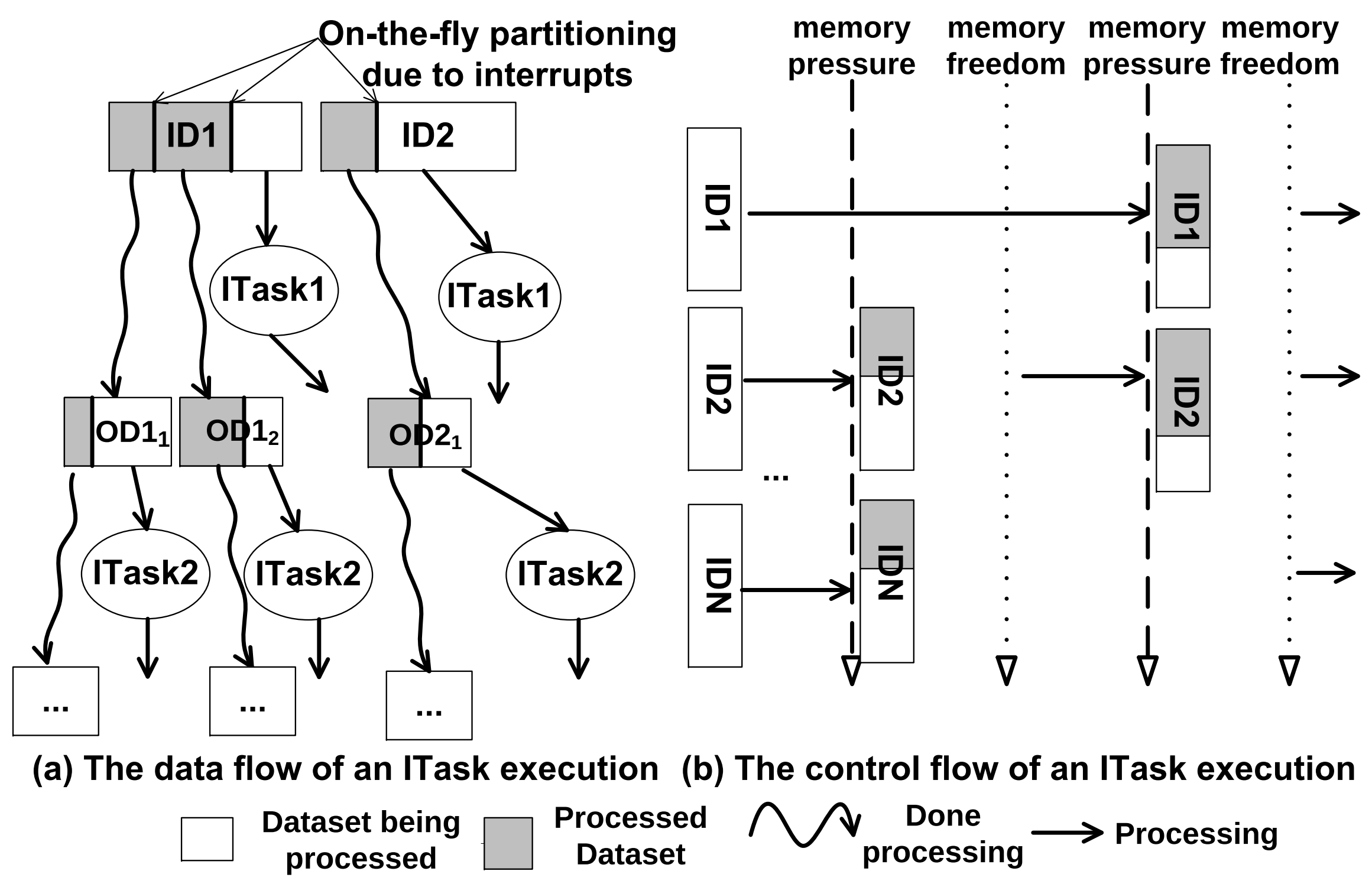
- Out of memory!!! Significant slow down!! Non-scalable!
- State-of-the-art frameworks
  - Hadoop [http://hadoop.apache.org]
  - Spark [Zaharia-NSDI'12]
  - Hive [Thusoo-ICDE'10]
  - Mahout [http://mahout.apache.org]
  - Pig [Olston-SIGMOD'08]
  - Hyracks [Borkar-ICDE'11]
- A common problem: memory pressure on single-node
  - An extensive study including 73 memory issues reported on StackOverflow [http://stackoverflow.com/]
  - Even using existing state-of-the-art automated tuning tools, e.g., YARN [Vavilapalli-SoCC'13], Mesos [Hindman-NSDI'11]
- Manual tuning is difficult!!!
  - Too many parameters, e.g., Hadoop has about 190 parameters
  - Requires highly-specialized experiences
  - Time consuming
  - Many problems cannot be solved by just tuning parameters

The key insights of ITask

- Main idea: *Treat memory pressure as interrupts*
  - A data parallel task can be interrupted upon memory pressure
  - An interrupted task can be resumed when memory pressure goes away
- *No need of*
  - Additional hardware resource
  - Manual parameter tuning

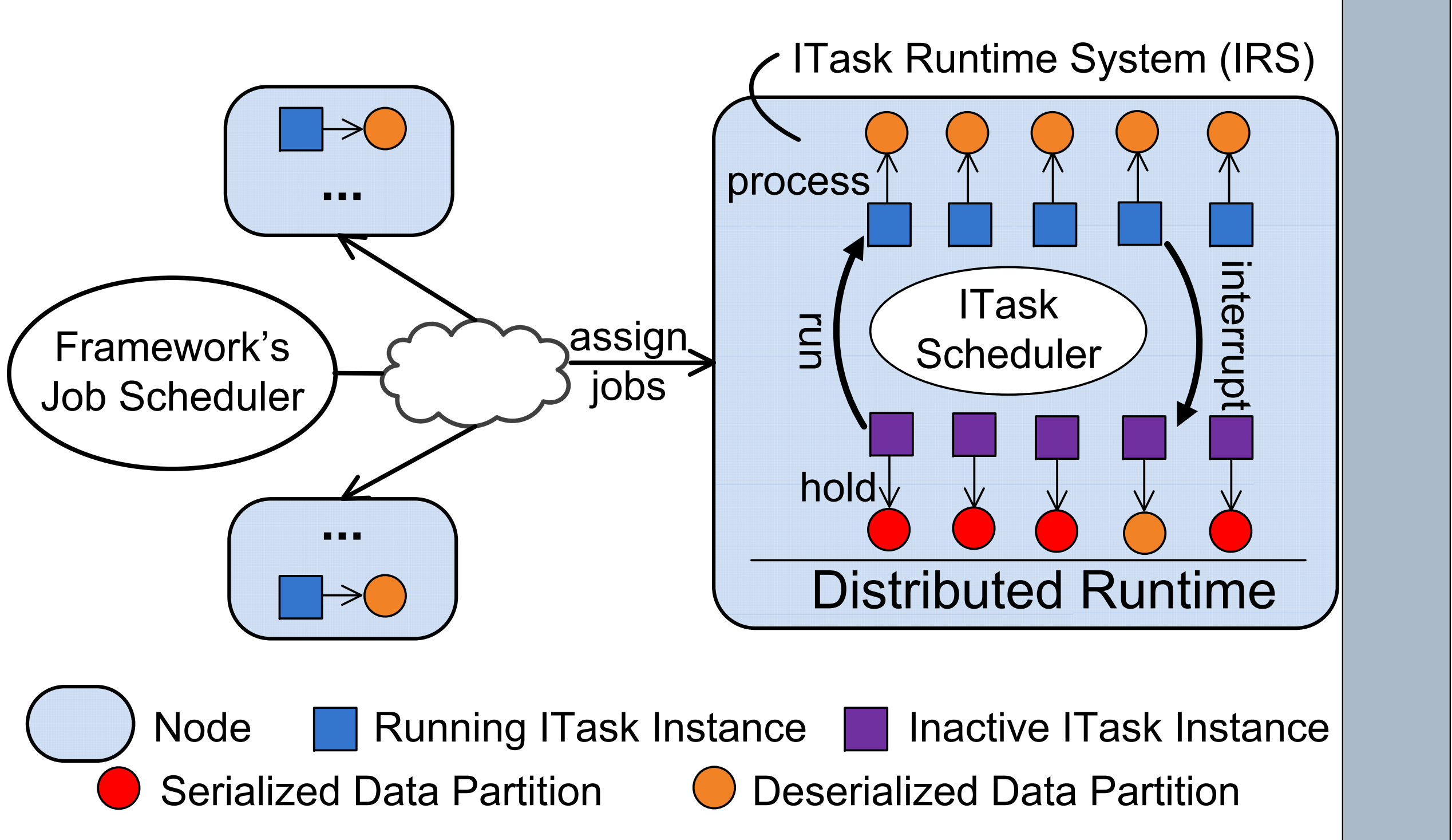
Novelties of ITask

- ITask works *proactively* in response to memory pressure
  - Take actions when a bellwether of memory pressure is seen
  - Take the system back to the memory usage "safe zone" even before much time is spent is spent on garbage collection (GC)
  - Improve both scalability and performance
- ITask uses *a staged approach* to lower its memory consumption
  - 5 stages: releasing (1) local variables, (2) the processed portion of the input, (3) partial output, (4) intermediate results, and (5) in-memory data, e.g., the rest of unprocessed data in memory
- *ITask is easy to implement*
  - ITask programming model: users (1) reconstruct code for existing tasks, (2) implement the abstract methods defined in ITask class
  - ITask runtime system: sits on top of existing frameworks, provides complementary optimizations and additional safety guarantee.



(a) The data flow of an ITask execution (b) The control flow of an ITask execution

The System Architecture



The ITask Programming Model

The ITask abstract class

- An existing task needs to *extend the ITask abstract class* to become an *interruptable* task.
- Four abstract functions are defined in ITask abstract class
  - *initialize, interrupt, cleanup, process*

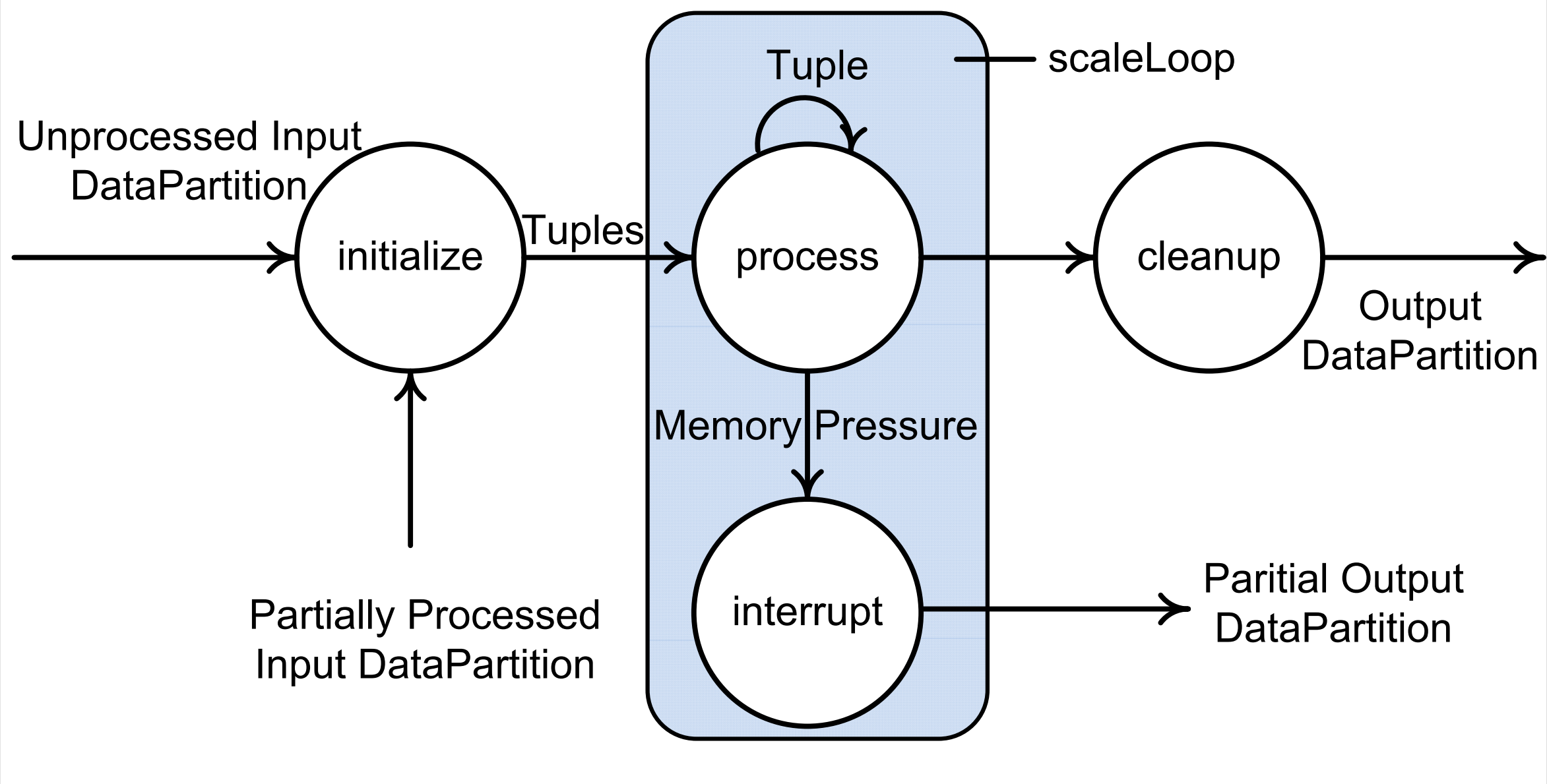
```
// The ITask abstract class in the library
abstract class ITask {
  abstract void initialize();   /* Initialization logic */
  abstract void interrupt();   /* Interrupt logic */
  abstract void cleanup();     /* Finalization logic */
  abstract void process(Tuple t);  /* Process a tuple */
  /* Scalable loop */
  boolean scaleLoop(DataPartition dp) {
    while (dp.hasNext()) {
      if (Monitor.hasMemoryPressure()
              && ITaskScheduler.terminate(this)) {
        /* Invoke the user-defined interrupt logic */
        interrupt();
        /* Push the partially processed input to the queue*/
        ITaskScheduler.pushToQueue(dp);
        return false;
      }
      process(dp.next());
    }
    return true;
  }
}
```

The ITask input and output

- Both input and output of an ITask are objects of type **DataPartition**
  - Developers only need to wrap an existing partition into a DataPartition Object
  - DataPartition: data tuples, a group tag, and a progress cursor

```
// The DataPartition abstract class in the library
abstract class DataPartition {
  /* The tag for grouping */
  int tag;
  /* The cursor points to the first unprocessed tuple */
  int cursor;
  /* Return whether there exists unprocessed tuple */
  abstract boolean hasNext();
  /* Serialize the DataPartition */
  abstract void serialize();
  /* Deserialize the DataPartition */
  abstract DataPartition deserialize();
}
```

The Execution of An ITask

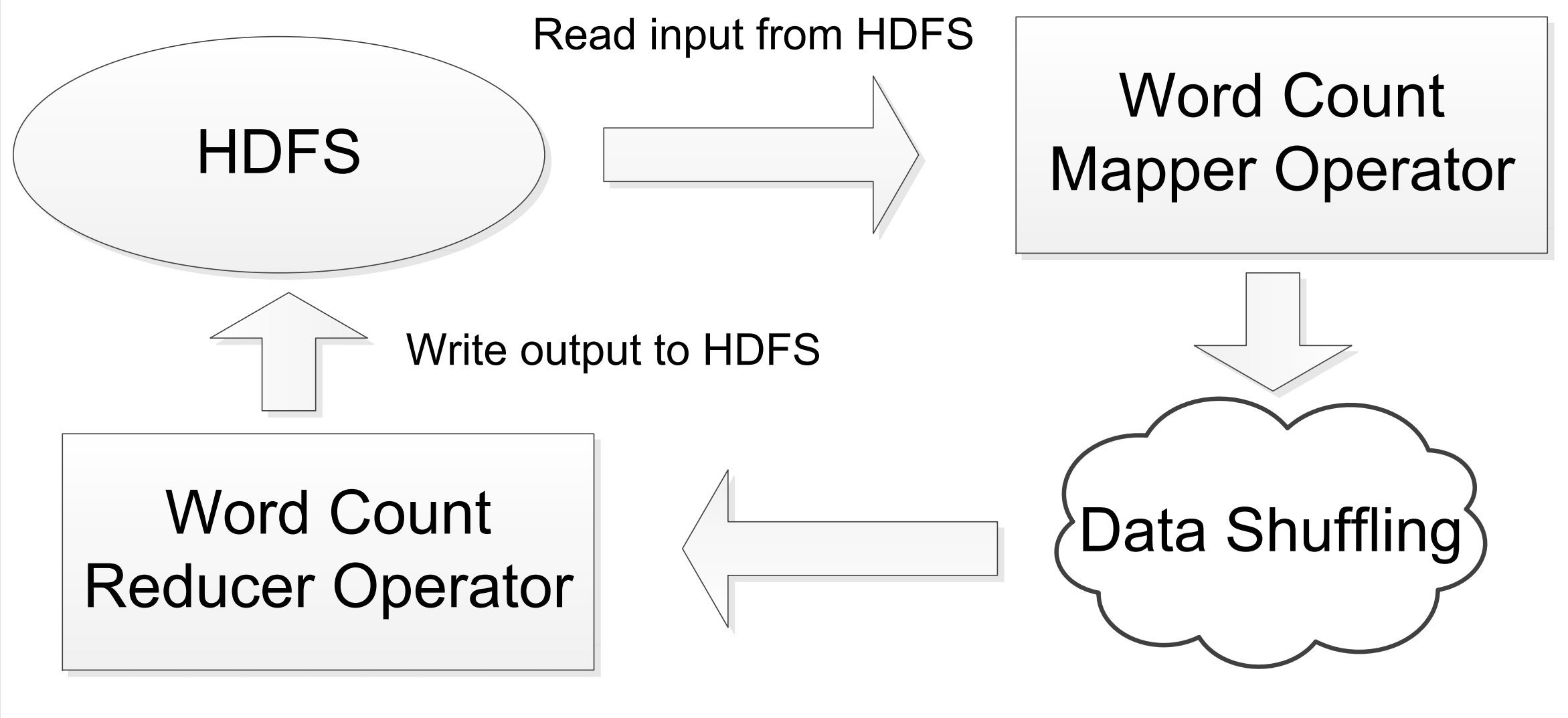

Tuple
scaleLoop
Unprocessed Input DataPartition
initialize
Tuples
process
cleanup
Output DataPartition
Memory Pressure
Partially Processed Input DataPartition
interrupt
Partial Output DataPartition

Instantiating ITasks in Existing Frameworks

On Hyracks

- A task in Hyracks is an implementation of HyracksOperator
  - HyracksOperator is an interface
  - An interruptable HyracksOperator needs to extend ITask

```
class MapOperator extends ITask implements HyracksOperator{
  MapPartition output;
  @Override
  void initialize() {
    /* Create an output partition */
    output = new MapPartition();
  }
  @Override
  void interrupt() {
    /* The output can be sent to reshuffling at any time */
    Hyracks.shuffle(output.getData());
    /* Release the processed parts of  the data partition */
    PartitionManager.release(output);
  }
  @Override
  void cleanup() {
    Hyracks.shuffle(output.getData());
  }
  @Override
  void process(Tuple t) {
    addWordInMap(output, t.getElement(0));
  }
  /* A function defined in HyracksOperator */
  void nextFrame(ByteBuffer frame) {
    /* Wrap the buffer into a partition object */
    BufferPartition b = new BufferPartition(frame);
    /* Set input and output */
    MapOperator.setInputType(BufferPartition.class);
    MapOperator.setOutputType(MapPartition.class);
    /* Push the partition to the queue and run ITask */
    ITaskScheduler.pushToQueue(b);
    ITaskScheduler.start();
  }
}
```



ITask WordCount Application on Hyracks

The ITask Runtime System

Monitor

- Send "Reduce" signal
  - When memory pressure is detected
- Send "Grow" signal
  - When the worker node has enough resource to start another thread

Partition Manager

- Serialize data partitions to disk
  - When memory pressure is detected (Receiving "Reduce" signal)
- Deserialize the data partitions from disk
  - When the data partitions are about to be processed

Scheduler

- Reduce the number of task instances
  - When memory pressure is detected and no more candidate partitions can be serialized to disk
- Create a new thread to run a task
  - When a "Grow" signal is received from the monitor

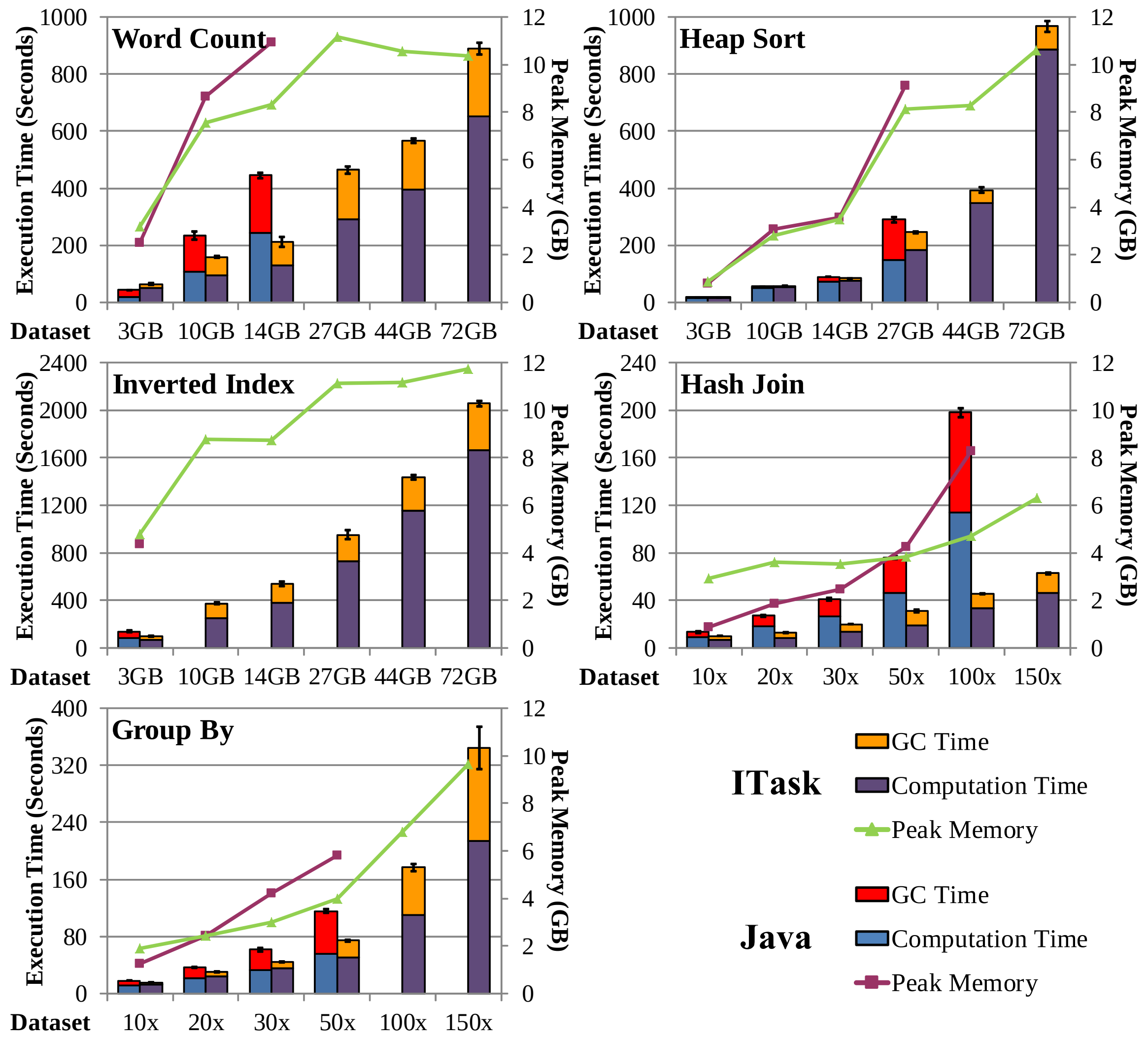
Implementation and Evaluation

ITask library implementation

- Hyracks 0.2.14 (newest version) [https://code.google.com/p/hyracks/]

Evaluation

- Datasets
  - *Yahoo Web Map*, for **WC**, **HS** and **II**
  - *TPC-H data*, for **HJ** and **GR**
- Performance improvements
  - The execution time is reduced 39.54%. (1.65x faster)
  - The peak memory consumption is reduced 9.26%.
  - The ITask programs can scale up to 24.00x larger datasets.



No data means the applications crash because of OutOfMemoryError

Conclusions

- ITask is the first attempt to help data-parallel tasks survive memory pressure and successfully scale to much larger datasets.
- It also relieves the system from high GC costs resulting from frequent useless and long GCs.
- ITask is a non-intrusive approach, and easy to use.