Interruptible Tasks: Treating Memory Pressure as Interrupts for Highly Scalable Data-Parallel Programs

Lu Fang[1], Khanh Nguyen[1], Guoqing Xu[1], Brian Demsky[1], and Shan Lu[2]

[1]University of California, Irvine
[2]University of Chicago

Motivation

- **Data-parallel system**
  - Input data is divided into *independent* partitions
  - Many popular big data systems
- **A common problem: *memory pressure on individual nodes***
  - Programs push the heap limit soon, and systems struggle for memory
  - Cause huge GC effort, badly hurt performance
  - Programs crash because of OutOfMemoryError
  - Many cases can be found on websites, such as StackOverflow
  - We have collected 126 problems by searching "out of memory" and "data parallel"
- **Root causes**
  - Hot keys
  - Large intermediate results
- **Existing solutions**
  - Configuration tuning
  - Skew fixing
  - Cluster-wide resource manager

We need a *systematic* solution for memory pressure on single nodes

Key Insights of Our Solution – ITask

- **Main idea: *treat memory pressure as interrupts***
  - Data-parallel tasks can be interrupted upon memory pressure
  - An interrupted task can be reactivated when memory pressure goes away
- **Original execution vs ITask execution**

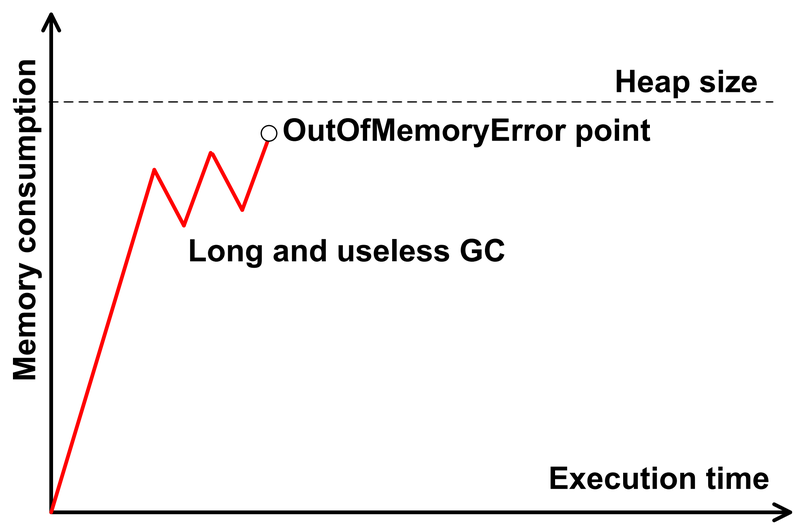

Fig. 1: Original data-parallel program suffers from memory pressure

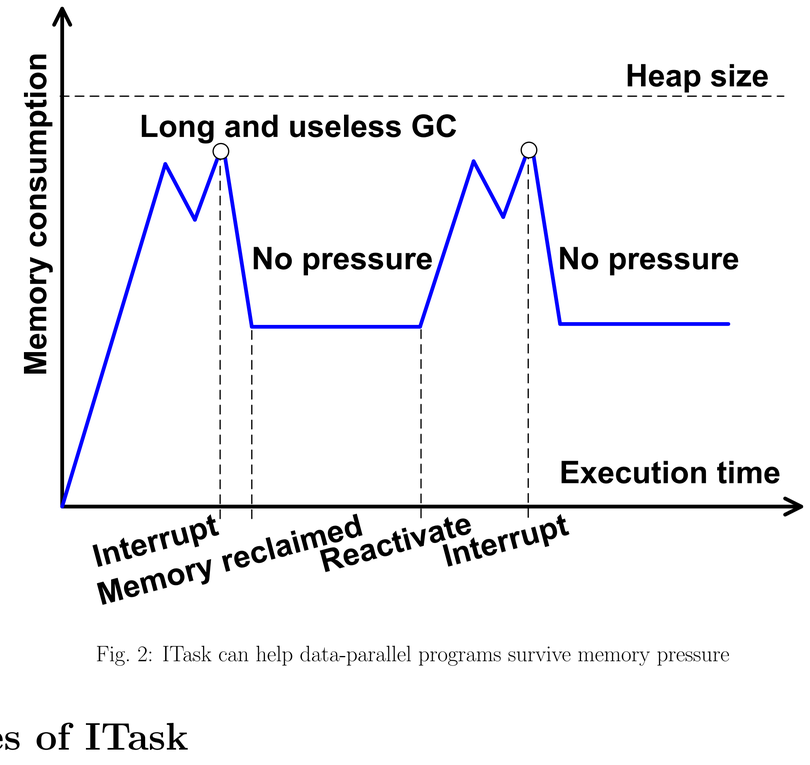

Fig. 2: ITask can help data-parallel programs survive memory pressure

- **Novelties of ITask**
  - ITask works *proactively* in response to memory pressure
  - ITask uses *a stage approach* to lower its memory consumption
  - ITask is *easy to apply* on existing frameworks

System Design

- **Challenges**
  - How to lower memory usage when a task is interrupted
  - When to interrupt a task
  - How to interrupt a task

Our approach consists of *a programming model* and *a runtime system*

- **Programming model**
  - API-based
  - Provide interrupt handling abstractions
- **ITask runtime system**
  - Monitor resource condition
  - Manage input and output data for ITasks
  - Schedule ITasks

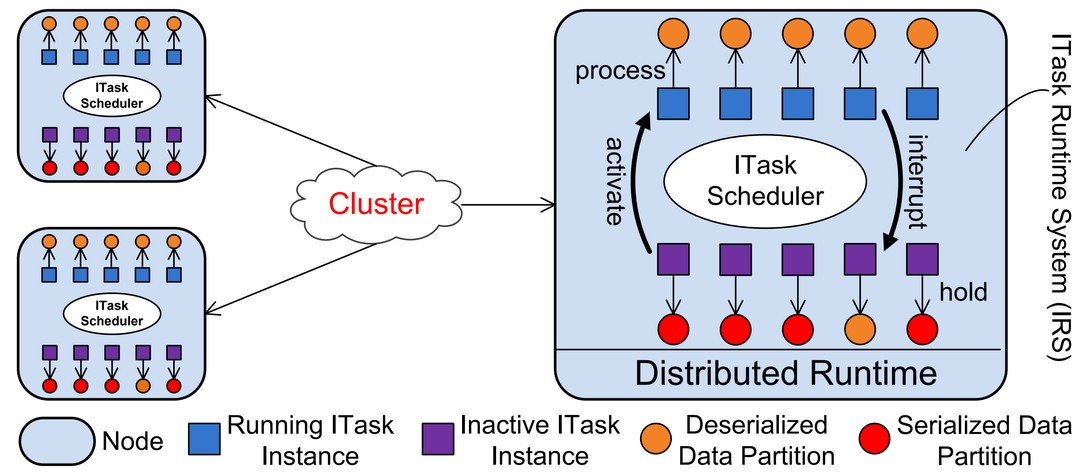

Fig. 3: The architecture of ITask runtime system

Evaluation

- **Environments**
  - We have applied ITask on two frameworks: *Hadoop 2.6.0* and *Hyracks 0.2.14*
  - An 11-node Amazon EC2 cluster
    - Each machine: 8 cores, 15 GB memory, 80×2 SSD RAID 0
- **Evaluation on Hadoop**
  - Goal: show the *effectiveness* of ITask on *real-world problems*
  - Benchmarks: five real-world programs collected from StackOverflow
    - Original version: crash because of OutOfMemoryError
    - Rfix version: apply the fixes recommended on the website
  - ITask helps all programs survive memory pressure
  - On average, ITask is *62.5% faster* than Rfix
- **Evaluation on Hyracks**
  - Goal: show the *improvements* of ITask on *performance* and *scalability*
  - Benchmarks: five already hand-optimized applications from Hyracks' code repository
  - On average, ITask is *34.4% faster* than original version
  - On average, ITask versions scale to *6.3× larger* datasets

Conclusions

- **ITask is the first systematic approach**
  - Help data-parallel tasks survive memory pressure
- **Design: a programming model + a runtime system**
  - Non-intrusive, easy to apply on existing systems
  - Easy to use
- **Evaluation shows the effectiveness of ITask system**
  - With ITask, real-world data-parallel programs survive memory pressure
  - ITask provides better performance than manually tuning configurations
  - ITask helps data-parallel tasks scale to larger datasets