On Fault Tolerance for Distributed Iterative Dataflow Processing

Abstract:

Distributed iterative processing is the preferred choice for large-scale graph and machine learning analytics on scale-out architectures. Naturally, this holds true because many graph and machine learning algorithms are iterative in nature and update their states (e.g., vertices and model parameters) at each iteration until convergence or a stopping criterion is satisfied. In order to address the need for large-scale distributed iterative processing, the information management community has developed a set of novel specially dedicated systems. For instance, Pregel, GraphLab and PowerGraph are designed for largescale distributed graph-parallel computation, whereas parameter servers and model selection management systems (MSMS) are designed for large-scale distributed machine learning. As a result, these dedicated systems enable a wide range of system optimizations by design (i.e., tailored specifically for use with either graph or machine learning analytics, respectively).

Existing System:

Large-scale graph and machine learning analytics widely employ distributed iterative processing. Typically, these analytics are a part of a comprehensive workflow, which includes data preparation, model building, and model evaluation. General-purpose distributed dataflow frameworks execute all steps of such workflows holistically. This holistic view enables these systems to reason about and automatically optimize the entire pipeline. Here, graph and machine learning analytics are known to incur a long runtime since they require multiple passes over the data until convergence is reached. Thus, fault tolerance and a fast-recovery from any intermittent failure is critical for efficient analysis.

Proposed System:

There is a need for efficient fault-tolerance mechanisms that reduce checkpointing costs and shorten recovery times for iterative algorithms for both graph processing and machine learning on distributed dataflow systems. For graph processing, in order to reduce the checkpoint writing overhead, we presented two checkpointing schemes: head and tail checkpointing. Unlike traditional approaches that manage checkpoints separately from the iteration pipeline, both of our strategies inject checkpoints into the execution pipeline, thus eliminating the need for an additional checkpoint manager. In order to accelerate the recovery phase, we introduced confined recovery with local logging during the group By stage in graph processing. For machine learning, we explored the use of a broadcast variable to achieve a fast recovery without relying on any checkpoints. The experimental and theoretical studies performed show that head checkpointing and confined recovery, as well as replica recovery for fault-tolerance on iterative processing outperforms blocking checkpointing and complete recovery.

Modules:

- Graph Processing
- Machine Learning

SYSTEM REQUIREMENTS

H/W System Configuration:-

Processor	-	Pentium –III
RAM	-	256 MB (min)
Hard Disk	-	20 GB
Key Board	-	Standard Windows Keyboard
Mouse	-	Two or Three Button Mouse
Monitor	-	SVGA

S/W System Configuration:-

Operating System	:	Windows95/98/2000/XP
Application Server	:	Tomcat5.0/6.X
Front End	:	HTML, Jsp

Scripts	:	JavaScript.
Server side Script	:	Java Server Pages.
Database	:	MySQL 5.0
Database Connectivity	:	JDBC