GroomServerFaultTolerance

GroomServerFaultTolerance (Draft)

Introduction

Distributed computing system such as [MapReduce][1], and Dryad[2] provide fault tolerance feature to help the system survive over the process crash. It is particular useful when computation requires to finish its execution in long time. Hama, based on the BSP[3] model, is a framework for massive scientific computations, which also requires this feature so that developers and users who exploit this framework can benefit from it. This page serves for providing information on direction how Hama GroomServer fault tolerance would work.

Literature Review

In general, a system designed to deal with failures usually need to apply techniques including unit of mitigation, redundancy, fault detection, fault recovery [4], and so on.

Unit of mitigation: GroomServer(s)/ BSPMaster

Redundant units: GroomServer(s)

Fault detection: System monitor, heartbeat.

Fault recovery: Fail over

Architecture

Task Failure

The execution of a task is spawned from the GroomServer so that the failure of the task would not pull down the GroomServer. Following steps are performed in the senario of task failure.

1. Whilst executing a task, the task will periodically ping its parent GroomServer.
2. If the GroomServer does not receive ping from the child (with timeout), it checks if child jvm is running; for instance, execute jps to identify child's status.
3. GroomServer notifies TaskScheduler that a task failure.
4. TaskScheduler updates JobInProgress.
5. TaskScheduler reschedules task to another GroomServer by searching an appropriate GroomServer.
6. If task rescheduled reaches the limit, the whole job fails.

GroomServer Failure

1. NodeManager embedded in the GroomServer periodically sends heartbeat to NodeMonitor in BSPMaster. Hama-370
2. One of GroomServers fails, indicating BSPMaster loses heartbeat from a particular GroomServer.
3. NodeMonitor Hama-363 collects metrics information, including CPU, memory, tasks, etc., from healthy NodeManagers.
4. Dispatch task(s) to GroomServer(s).
    a. NodeMonitor notifies TaskScheduler the failure of GroomServers; and move failure GroomServer to black list (will move back when the failed GroomServer restarts).
    b. TaskScheduler searches node list looking for GroomServer(s) whose workload is not heavy (which GroomServer to go is corresponded to policy).
    c. Update task(s) JobInProgress by assigning failed tasks to the GroomServer found in previous step.
    d. Dispatch task(s) to designed GroomServer(s).

Glossary

NodeMonitor: a component monitors the healthy of GroomServers.

NodeManager: a component that collects metrics information whilst NodeMonitor requests to report status of the GroomServer it runs on.

References

[1]. [MapReduce]: simplified data processing on large clusters. http://portal.acm.org/citation.cfm?id=1327492

[2]. Dryad: distributed data-parallel programs from sequential building blocks. http://portal.acm.org/citation.cfm?id=1273005

[3]. Bulk Synchronous Parallel Computing – A Paradigm for Transportable Software. http://portal.acm.org/citation.cfm?id=798134

[4]. Patterns for Fault Tolerant Software. http://portal.acm.org/citation.cfm?id=1557393

[5]. Supervisor Behaviour. http://www.erlang.org/doc/design_principles/sup_princ.html

[6]. Extensible Resource Management For Cluster Computing. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=603418