Dryad

Distributed Data Parallel Programs from Sequential Building Blocks

Yahn-Chung (Andrew) Chen
Overview

• Design Goals

• The Design of Dryad

• Discussion & QA
Problem

• How can we make it easier for developers to write efficient parallel and distributed application?
Problem

• How can we make it easier for developers to write efficient parallel and distributed application?

• Developers no need to have understanding in standard concurrency mechanism. (No OS course in the future)
Dryad goal

- General-purpose execution environment for distributed, data-parallel applications
  - Focus on throughput than latency

- Automatic management for scheduling, distribution, fault tolerance, etc.
What’s Dryad?

• Many programs can be represented as a distributed execution graph

• Dryad is the middleware abstraction run for you
  - Dryad sees arbitrary graphs
  - Above Dryad is just graph manipulation


Directed Acyclic Graph

Input

Output
Directed Acyclic Graph

Input

Processing vertices

Output
Directed Acyclic Graph

Input

Output

Channel
- Disk
- TCP
- Memory

Processing vertices
Channel Types

• Sequence of structured items

• Implementation
  - Temporary Disk file
    - Item are serialized in buffer
  - TCP pipe
    - Item are serialized in buffer
  - Shared-memory FIFO
    - Pass pointer to items directly
Directed Acyclic Graph

Directed cyclic graph

Input

Processing vertices

Output

Channel
- Disk
- TCP
- Memory
Job execution

- **Scheduler** in Job Manager

- Exec of vertex
  - Version number
  - Exec record
  - State
  - Predecessor’s version
Job execution

- Scheduler in Job Manager
- Exec of vertex
  - Version number
  - Exec record
  - State
  - Predecessor’s version

Put into scheduling queue
Job execution

- Scheduler in Job Manager

- Exec of vertex

  - Version number
  - Exec record
  - State
  - Predecessor’s version

  Periodically communicate with JM
Job execution

- Scheduler in Job Manager

- Exec of vertex
  - Version number
  - Exec record
  - State
  - Predecessor’s version

File-based output channel with version name
Job execution

• Scheduler in Job Manager

• Exec of vertex
  - Version number
  - Exec record
  - File-based output channel with version name

  State
  - Predecessor’s version
  - Preference (optional)
Fault tolerance

- Vertex error $\rightarrow$ Job manager
- Process crashes $\rightarrow$ Job manager
- Daemon fail $\rightarrow$ Job manager receive timeout (heartbeat)
- Read Error on input channel $\rightarrow$ Mark the channel as failed
Stage manager

• Each vertex belongs to a stage

• Report statistics

• Get callback on interesting events
  - Gather execution statistics
  - Request duplicate executions
Connection Manager

• Any pair of stages can be linked

• Get callback on interesting events
  In upstream stage
    - For dynamic optimization
Aggregation Manager

• Graph Refinement

Data reduction

Insert new layer

Reduce network traffic
Example

- SkyServer DB Query
  Table U (objId, color)
  Table N (objId, neighborId)

Find neighbor star with similar colors
Join U + N to find
  
  T = U.color, N.neighborID WHERE U.objId = N.objId

Join U + T to find
  
  U.objId WHERE U.objId = T.objId AND U.color = T.color
Example

• SkyServer DB Query
  Table U (objId, color)
  Table N (objId, neighborId)

Find neighbor star with similar colors
Join U + N to find
  T = U.color, N.neighborId WHERE U.objId
Join U + T to find
  U.objId WHERE U.objId = T.objId AND
Example

Example started from (21:46):
https://www.youtube.com/watch?v=WPhE5JCP2Ak