TASK SCHEDULING

Task Scheduling Definition

- Task scheduling is the process of scheduling multiple task instances into the cluster resources which mainly aims to minimize the task completion time and increase the utilization of resources.
- The main objective is scheduling the task instances in an appropriate order to suitable/appropriate cluster nodes which are classified into two levels namely Task Selection and Node Selection.
- In the task selection process, task instances are prioritized in an order (considering the dependency also) for the execution,
- In the node selection process selecting the appropriate nodes for the execution of those prioritized tasks.

Task Scheduling Definition

- Other main functional requirements of task scheduling are;
  - Scalability
  - Dynamism
  - time and cost efficiency
  - handling different types of processing models, data and jobs, etc.
- Other major objectives of task scheduling are;
  - reducing the number of task migrations
  - allocating the number of dependent and independent tasks in a near optimal manner to;
    - decrease the overall computation time of a job
    - improve the utilization of cluster resources effectively.

Classification of Task Scheduling Systems

- Task scheduling algorithms are broadly classified into two types;
  - Static
  - Dynamic
- Static task scheduling algorithms
  - jobs are allocated to the nodes before the execution of a job and the processing nodes are known at compile time.
  - Once the tasks are assigned to the appropriate resources, the execution continues to run until task completion.
  - The main objective of the static task scheduling strategy is to reduce the scheduling overhead that occurs during the runtime and minimize the number of nodes/processors.
  - Capacity Scheduling, Data Locality-Aware Scheduling, Round Robin Scheduling, Delay Scheduling, FIFO Scheduling, First Fit Scheduling, Fair Scheduling, Matchmaking Scheduling, and so on are some of the examples of static task scheduling algorithms.

Classification of Task Scheduling Systems

- Dynamic task scheduling algorithms
  - Dynamic task scheduling takes the scheduling decisions during the runtime of task execution.
  - It mainly considers the resource requirement, availability of resources, interprocess and inter-node traffic, energy efficiency, and more.
  - It also supports task migration which is based on the status of the cluster resources and the workload of an application.
  - Some of the most popular dynamic task scheduling examples are resource-aware scheduling, energy-aware scheduling and deadline-aware scheduling.

Task Scheduling considerations for Batch Processing

- In general, task scheduling for batch jobs can be performed prior to processing, based on the knowledge of input data and task information for processing in a distributed environment.
    - In addition, the resources can be statically allocated prior to the execution of a job.
- Florin Pop and Valentin Cristea explained processing of big data as a big batch process by splitting a job into multiple tasks and running on a High Performance Computing (HPC) by distributing the work to the cluster nodes.

Task Scheduling considerations for Streaming Processing

- In general, the big data platform receives a large amount of streaming data from input data streams such as data sensors, social networking, IoT devices and others.
- It is difficult to store such large amounts of streaming data hence, it should be processed immediately which requires a lot of computation cycles and memory resources.
- The scheduling for streaming mainly focus on minimizing latency.

Task Scheduling in Popular Big Data Tools or Platforms

- Apache Hadoop
  - Implemented with the default scheduling policy of FIFO which schedules the jobs coming first and gets higher priority than the later one that leads to starvation of jobs.
- Apache Spark
  - It mainly works on the principle of Resilient Distributed Datasets (RDDs) which has been implemented both the static and dynamic task scheduling algorithms for those RDDs.
  - The fair scheduling policy in Spark group the jobs into pools and assign weights into each pool.
  - The dynamic resource allocation policy allocates the resources to the jobs based on the workload of the cluster resources in a dynamic manner.
- Apache Mesos
  - It is implemented with a fine-grained Dominant Resource Fairness (DRF) algorithm that allocates the sharing of resources across the applications running on the platform.
  - It decides a number of resources to be allocated to each framework and provides the resource offers to the schedulers.

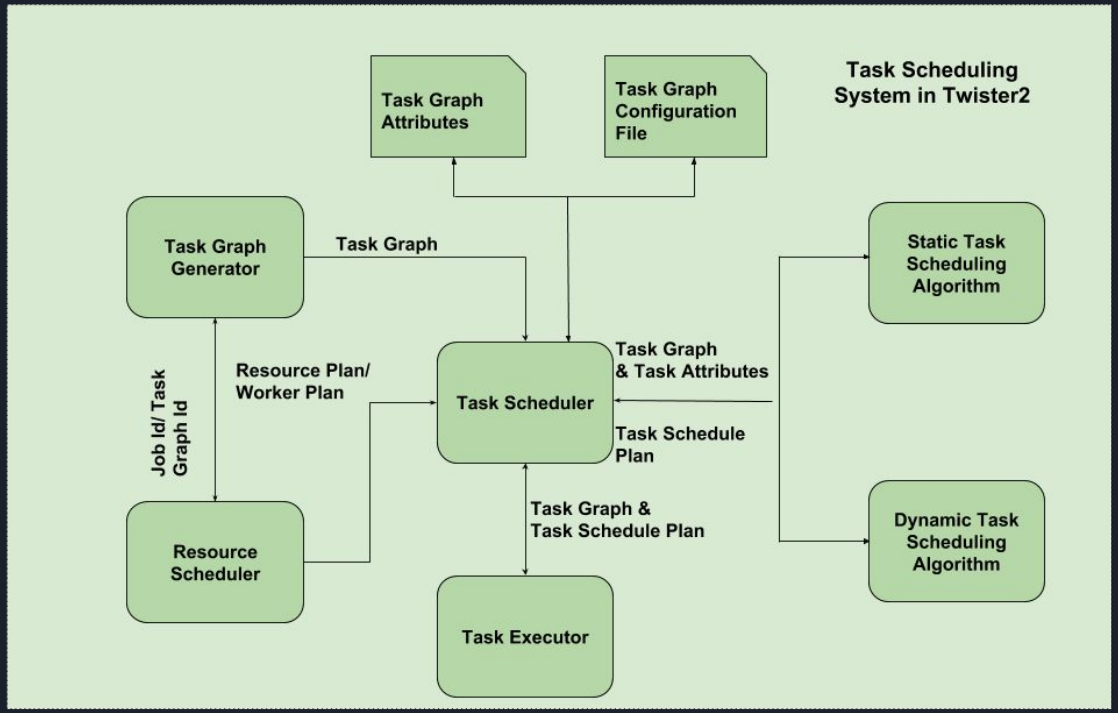
Task Scheduling in Popular Big Data Tools or Platforms (Contd.)

- Apache Flink
  - It is implemented with an immediate scheduling and queued scheduling algorithm which returns the slot immediately and queues the request and returns the slot whenever it is available respectively.
- Apache Storm
  - Default round-robin scheduling for the placement of streaming tasks on the execution nodes.
  - Doesn’t consider the availability of the resource or the applications resource requirement while scheduling the tasks, this may lead to under-utilization or over-utilization of the resources.
- Apache Heron
  - Implemented with Round Robin and First Fit Bin Decreasing packing plan algorithms for scheduling the streaming applications in the big data processing.
  - Similar to Apache Storm, by default, Apache Heron doesn't consider the resource requirement and the resource availability which leads to under-utilization or over-utilization of resources.

Task Scheduling in Twister2

- In Twister2, a job or an application is represented as a dataflow programming model in which tasks are organized in a graph structure.
- Scheduling these tasks is one of the key active research areas which mainly aims to place the tasks on available resources.
- It is essential to effectively schedule the tasks, in a manner that minimizes task completion time and increases utilization of resources.
- The selection of the best method is a major challenge in the big data processing environment.

Task Scheduling Architecture in Twister2



Scheduling Model in Twister2

The scheduling in twister2 comprises of;

- **Job model** - It provides the abstraction of jobs (consists of multiple tasks) and its requirements. The requirements are classified into hard and soft constraints. The job model handles different type of jobs namely batch, streaming, MPI, and microservices.
- **Resource model** - It describes the characteristics and performances of data centers, hosts, rack, and network links.
- **Scheduling policy/algorithm** - The scheduling policy or algorithm implemented in Task Scheduler which is based on specific goals such as optimization of total computational time or utilization of cluster resources or both.
- **Performance metrics** - It is used to evaluate the performance improvements gained by the proposed task scheduling model (for ex: data locality)
- **Programming model** - The programming model is helpful for providing the interface to the scheduler. In this proposed approach, we have made use of dataflow programming model for interfacing the task scheduler with other components.

Task Scheduler in Twister2

- The task scheduler in twister2 is designed in a way such that it is able to handle both streaming and batch tasks. It supports the following task scheduling algorithms namely Round Robin, First Fit, and Data Locality.

Task Scheduler Name	Container Allocation Types	Task Instance Types
Round Robin Task Scheduler	Homogeneous	Homogeneous
First Fit Task Scheduler	Heterogeneous	Heterogeneous
Data Locality Task Scheduler	Homogeneous	Homogeneous
Resource-Constraint Round Robin Task Scheduler	Heterogeneous	Heterogeneous

Round Robin Task Scheduler

- RoundRobin Task Scheduler allocates the task instances of the task graph in a round robin fashion. For example;
  - if there are 2 containers and 2 tasks with a task parallelism value of 2, task instance 0 of 1st task will go to container 1 and task instance 0 of 2nd task will go to container 1 whereas task instance 1 of 1st task will go to container 1 and task instance 1 of 2nd task will go to container 2.
- It generates the task schedule plan which consists of the containers (container plan) and the allocation of task instances (task instance plan) on those containers. The size of the container (memory, disk, and cpu) and the task instances (memory, disk, and cpu) are homogeneous in nature.
- First, it will allocate the task instances into the logical container values and then it will calculate the required ram, disk, and cpu values for the task instances and the logical containers which is based on the task configuration values and the allocated worker values respectively.

First Fit Task Scheduler

- FirstFit Task Scheduler allocates the task instances of the task graph in a heuristic manner. The main objective of the task scheduler is to reduce the total number of containers.
- It generates the task schedule plan which consists of the containers (container plan) and the allocation of task instances (task instance plan) on those containers. The size of the container (memory, disk, and cpu) and the task instances (memory, disk, and cpu) are homogeneous in nature.
- For example;
    - if there are two tasks with parallelism value of 2, 1st task -> instance 0 will go to container 0, 1st task -> instance 1 will go to container 0, 2nd task -> instance 0 will go to container 0 until the total instance required values reach the maximum size of the container 0. Once the container has reached its maximum limit then it will allocate the 2nd task -> instance 1 will go to container 1.

Data Locality Task Scheduler

- Data Locality Aware Task Scheduler allocates the task instances of the streaming task graph based on the locality of data.
- It calculates the distance between the worker nodes and the data nodes and allocate the streaming task instances to the worker nodes which are closer to the data nodes i.e. it takes lesser time to transfer/access the input data file.
- The data transfer time is calculated based on the network parameters such as bandwidth, latency, and size of the input file.
- It generates the task schedule plan which consists of the containers (container plan) and the allocation of task instances (task instance plan) on those containers.

Comparison With Other Systems

Big Data Platforms	Scheduling Types		Scheduling Job Types				Dataflow Programming Model
	Static	Dynamic	Batch	Streaming	MPI	FaaS/ Microservices	
Spark	No	Yes	Yes	Yes	No	No	Yes
Flink	Yes	No	Yes	Yes	No	No	Yes
Heron	Yes	No	No	Yes	No	No	Yes
Storm	Yes	No	No	Yes	No	No	Yes
Hadoop	Yes	No	Yes	No	No	No	No
Twister2	Yes	Yes	Yes	Yes	Yes	No (Future Work)	Yes

Conclusion

- Task scheduling in Big data is one of the active research areas which plays a major role in the completion of Big data processing and effectively utilize the cluster resources.
- Task scheduling in Twister2 has the common task scheduling model to accommodate both the static and dynamic task graphs.
- Additionally, it has the ability to schedule both streaming and batch task graphs.