List Scheduling

masrur-jamil.prochchhod

June 2024

Abstract

List Scheduling is one of these techniques that is particularly useful and flexible for coordinating work execution across many computing resources. This paper provides a thorough introduction to List Scheduling techniques, covering algorithmic complexities, theoretical underpinnings, empirical assessments, and practical applications. List scheduling works by keeping track of a list of activities that are prioritised and dynamically assigning resources to optimise resource utilisation, minimise delay, and increase system throughput. This paper investigates the diverse field of List Scheduling research by utilising theoretical analyses, innovative algorithms, and real-world implementations. Additionally, the use of list scheduling in high-performance computing, real-time systems, cloud computing, and parallel processing is examined in this work. This paper intends to promote innovation in parallel and distributed computing paradigms and develop task scheduling approaches by synthesising existing research findings and emphasising new trends and obstacles.

Keywords- Heuristic; task dependency; scalability; performance evaluation;

1 Introduction

List Scheduling is a critical component in the realm of task scheduling algorithms, particularly in the context of parallel and distributed computing systems. Essentially, it functions as a key tactic for effectively coordinating the execution of activities across several computing resources. As computing systems grow in complexity and size, the significance of using the best work scheduling techniques becomes clearer.

The goal behind list scheduling is to keep a prioritised list of tasks that need to be completed and to dynamically allocate available resources in order to maximise system throughput, reduce latency, and optimise resource utilisation. In contrast to conventional scheduling methods, which could concentrate on particular metrics or restrictions, like minimising make span or fulfilling deadlines, list scheduling offers a versatile framework adaptable to diverse computing environments and objectives.

Fundamentally, list Scheduling relies on algorithms or heuristics that are intended to make immediate choices about allocating tasks to available resources. Task dependencies, resource availability, execution time estimations, and system limits are a few examples of the variables that frequently influence these choices. List Scheduling attempts to balance conflicting goals by constantly updating and rearranging the task list in response to shifting system states, with the ultimate goal of improving system performance and efficiency.

The field of list Scheduling study is broad and includes theoretical studies, algorithmic breakthroughs, empirical assessments, and real-world implementations in a range of industries. Researchers examine the complexities involved in developing effective scheduling algorithms, weighing trade-offs between various performance measures, and coming up with solutions for scaling issues in large-scale computer settings.

Additionally, list scheduling is used in many different fields, such as real-time systems, grid computing, cloud computing, high-performance computing (HPC), parallel processing, and grid computing. Because of its scalability and agility, it may be used in a wide range of applications, from mission-critical operations and industrial automation to data analytics and scientific simulations.

The paper attempts to give a thorough introduction to list scheduling approaches, exploring its theoretical underpinnings, algorithmic complexities, empirical assessments, and practical applications. This work aims to promote innovation in parallel and distributed computing paradigms and develop task scheduling approaches by synthesising existing research findings and illuminating new trends and obstacles.

2 Example of List Scheduling Algorithm

2.1 Tasks and Processors

- Tasks: $T = \{T_1, T_2, T_3\}$
- Processors: $P = \{P_1, P_2\}$
- Durations: $d_1 = 3$, $d_2 = 2$, $d_3 = 4$
- Dependencies: $dependencies(T_3) = \{T_1, T_2\}$, $dependencies(T_1) = \emptyset$, $dependencies(T_2) = \emptyset$
- Priorities: $priority(T_1) = 1$, $priority(T_2) = 2$, $priority(T_3) = 3$

2.2 Scheduling Steps

1. **Initialization**:

- Priority list $L = \{T_3, T_2, T_1\}$
- $available\_time(P_1) = 0$
- $available\_time(P_2) = 0$

2. **Schedule Tasks**:

- **Task** T_3:
  - Calculate earliest start time:

$$earliest\_start(T_3) = \max(finish(T_1), finish(T_2)) = 0$$

  - Assign T_3 to P_1:

$$start(T_3) = 0, \quad finish(T_3) = 0 + 4 = 4$$

  - Update $available\_time(P_1) = 4$

- **Task** T_2:
  - Calculate earliest start time:

$$earliest\_start(T_2) = 0$$

  - Assign T_2 to P_2:

$$start(T_2) = 0, \quad finish(T_2) = 0 + 2 = 2$$

  - Update $available\_time(P_2) = 2$

- **Task** T_1:
  - Calculate earliest start time:

$$earliest\_start(T_1) = 0$$

  - Assign T_1 to P_2:

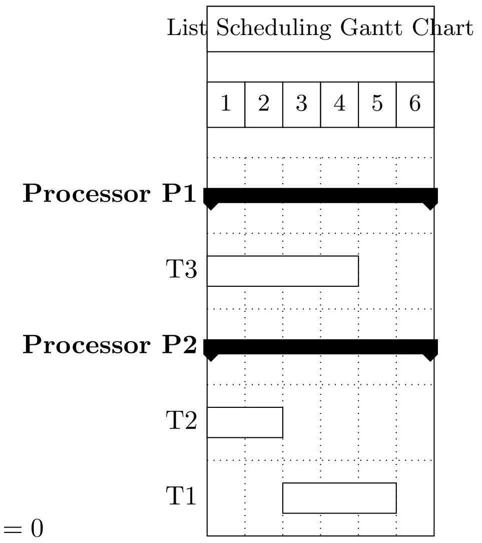
$$start(T_1) = 2, \quad finish(T_1) = 2 + 3 = 5$$

  - Update $available\_time(P_2) = 5$

3. **Output Schedule**:

$$S = \{(T_3, P_1, 0, 4), (T_2, P_2, 0, 2), (T_1, P_2, 2, 5)\}$$

2.3 Gantt Chart Visualization



This Gantt chart visually represents the schedule:

- **Processor P1**: Task T3 runs from time 0 to 4.
- **Processor P2**: Task T2 runs from time 0 to 2, and Task T1 runs from time 2 to 5.

This simple example demonstrates how List Scheduling allocates tasks to processors to optimize their execution based on priorities and dependencies.