AN OVERVIEW OF THE SCHEDULING POLICIES AND ALGORITHMS IN GRID COMPUTING

D.I. George Amalarethinam, Director-MCA & Associate Professor of Computer Science, Jamal Mohamed College (Autonomous), Tiruchirappalli, India. **P. Muthulakshmi**, Lecturer, Department of Computer Science, SRM University, Kattankulathur, Chennai, India

Presented by: Dr. Faramarz Safi

Islamic Azad University, Najafabad Branch,

Esfahan, Iran.

Abstract

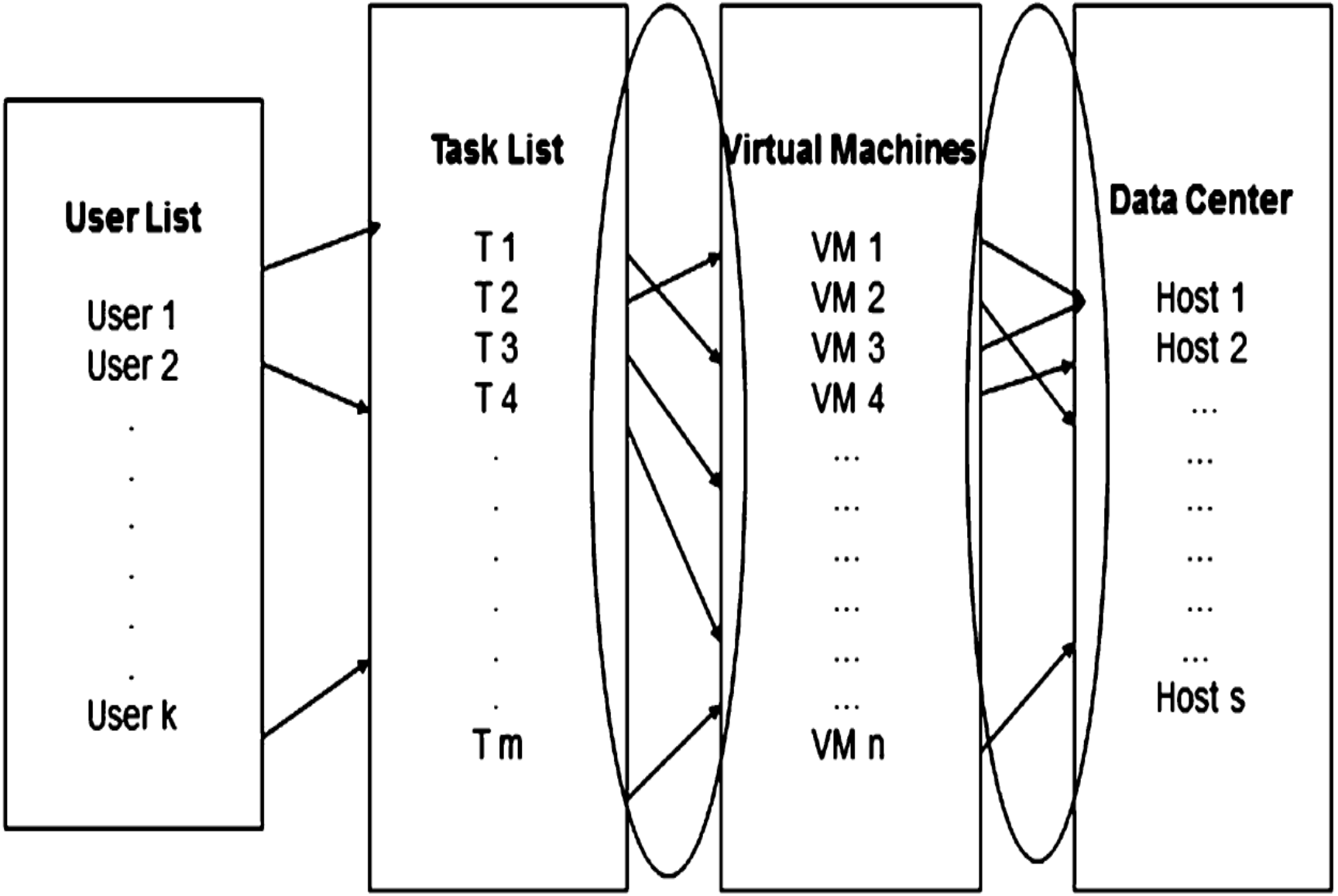
- Scheduling is the ultimate goal aiming process of mapping the jobs submitted to the cloud or grid and the appropriate resources available.

- This paper presents a package of reviews taking **various factors** that are having greater influence while scheduling the jobs.

- Here we consider the algorithms keyed with **communication cost**, **execution time length**, **error factors**, **task duplications**, **data intensive**, and **heterogenic network behavior** and so on.

Introduction

- Foster et al [1] defined **Grid computing** as a "**coordinated resource sharing and problem solving, dynamic, multi-institutional collaborations**".
- The novel idea behind the technology is to identify the resources in the network of computers that are geographically spread and allow the computers to coordinate themselves in sharing the resources to solve the complex problems in short span of time and at economical rate.
- **Scheduling** refers to **the way processes are assigned to run on the available processors**. A **task (job) scheduling** is a process of **establishing queue to run a sequence of programs over a period of time**.
- A **task scheduling** is the **mapping of tasks to a selected group of resources, which may be distributed among multiple administration domains**.
- T**he problems lie in developing an algorithm to suit the various forms of applications**, including, **process of partitioning the application into tasks**, **encouraging coordinated communication among tasks**, **monitoring the synchronization among tasks**, **and mapping (scheduling) the tasks to the machines**.


User List
User 1
User 2
User k
Task List
T 1
T 2
T 3
T 4
T m
Virtual Machines
VM 1
VM 2
VM 3
VM 4
VM n
Data Center
Host 1
Host 2
Host s

Introduction

- The core objective of scheduling is to minimize the completion time of a **parallel application** by properly and carefully allocating the tasks to the appropriate processors.
- Scheduling can be classified into two main root categories called **static** and **dynamic**.
- In static scheduling, there is no job failure and resources are assumed available all the time. But this is not so in dynamic scheduling.
- There are essentially two main aspects that determine the dynamics of the scheduling, namely:

(a) **the dynamics of job execution**, which refers to the situation when job execution could fail due to some resource failure or job execution could be stopped due to the arrival in the system of high priority jobs when the case of preemptive mode is considered; and

(b) **the dynamics of resources**, in which resources can join or leave the Grid in an unpredictable way, their workload can significantly vary over time, the local policies on usage of resources could change over time, etc.

These two factors decide the behavior of the Grid Scheduler (GS), ranging from static to highly dynamic.

Scheduling in Grid

- **Foundation/fabrication level**, where the physical components are finding part, next sits.
- **Middleware**, which is actually the software, responsible for resource management, task execution, job scheduling,
- **Vendors/users**, who is going to use the efficiency of the grid solutions
- Finally the level with the services such as operational utilities and
- **Business solutions/tools**, which will be made use by the users of the grid, that we can bind the services and the applications.

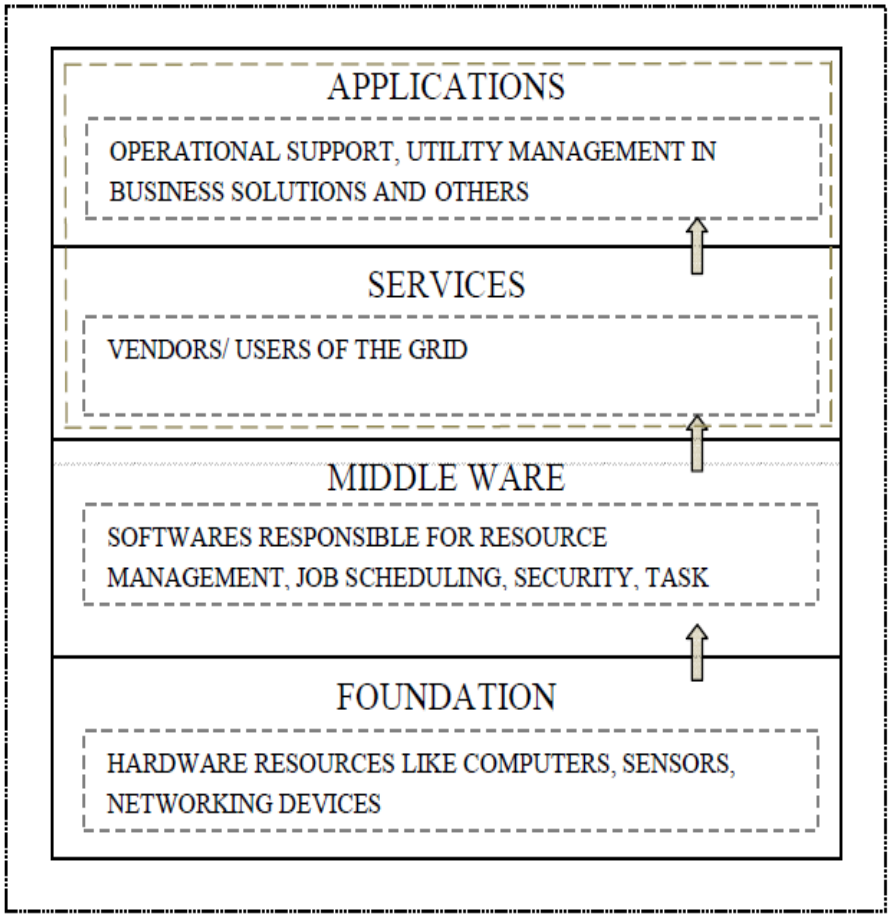


Figure1. Taxonomy of grid

Scheduling

- The **primary aim of grid** is to provide transparent and efficient access to remote and geographically distributed resources. A scheduling service is needed to coordinate the access to the different resources.
- Due to the dynamicity and heterogeneity, scheduling jobs is a bit complicated matter. **Jobs queuing and dispatching the jobs to the available nodes (processors) are the works of the scheduler**.

- Grid schedulers dispatch jobs following the eminent algorithms like, **First Come First Serve algorithm**, **Shortest Path First algorithm**, **Round Robin technique**, **Least Recently Serviced**.
- Scheduling is done on taking the factors like **job priority**, **reservation of resources**, **specific resource requirement of the job**, etc, **Keeping check points and mentoring** is a great mechanism, whereby places at which the process stopped or malfunctioned can be identified and the process can restart there at right for further completion. The advantage of keeping check points is that, the stopped process can resume from the place at which it stopped rather starting from the initial stage of the process. Fault due to load imbalance and other cause results in the migration jobs to other node. As it is the decentralized system the jobs can be sent and received by the local schedulers, failure detection of any node may not affect the computation and a simultaneous rescheduling is needed.

- Some of the eminent factors influencing the job scheduling: **Dependency among tasks**, **communication cost**, **execution cost**, **error factors like node failure**, **network failure**, **data storage**, **task duplications**, **execution time length**, and **heterogenic network behavior**, **fault tolerance**, **dynamicity, performance**, **cost efficiency**, **Quality of Service**, **time dependency**, **resource recovery**, **resource allocation**, and **management**, **security** so on.

Scheduling policies

- **Immediate Mode:**

In immediate mode scheduling, tasks are scheduled as soon as they enter the system. They will not be waiting for the next time interval. They will be scheduled as when the scheduler will get activated or the job arrival rate is small, having thus available resources to execute jobs immediately.

- **Batch Mode:**

In batch mode scheduling, tasks are grouped into batches which are allocated to the resources by the scheduler. The results of processing are usually obtained at a later time. Batch scheduling could take better advantage of job and resource characteristics in deciding which job to map to which resource since they dispose of the time interval between two successive activations of the scheduler.

- **Preemptive Scheduling**

In the preemptive mode, preemption is allowed; that is, the current execution of the job can be interrupted and the job is migrated to another resource. Preemption can be useful if job priority is to be considered as one of the constraints.

Scheduling policies

- **Non-preemptive Scheduling**

In the non-preemptive mode, a task, job or application should entirely be completed in the resource (the resource cannot be taken away from the task, job or application).

- **Static Scheduling**

In static scheduling, the information regarding all the resources in the Grid as well as all the tasks in an application are assumed to be known in advance by the time the application is scheduled and further more a task is assigned once to a resource.

- **Dynamic Scheduling**

In dynamic scheduling, the task allocation is done on the go as the application executes, where it is not possible to find the execution time. The jobs are entering dynamically and the scheduler has to work hard in decision making to allocate resources. This is the place where the load balancing is a factor to be considered seriously. The advantage of the dynamic over the static scheduling is that the system need not possess the runtime behavior of the application before it runs.

- **Independent Scheduling**

In independent scheduling, the applications are written to be able to be partitioned into almost independent parts (or loosely coupled), which can be scheduled independently.

Scheduling policies

- **Centralized Scheduling**

In dynamic scheduling scenarios, the responsibility for making global scheduling decisions may lie with one centralized scheduler, or may be shared by multiple distributed schedulers. The **advantage** of having centralized scheduling is the **ease of implementation**, but **lacks scalability, fault tolerance and sometimes performance**. In centralized scheduling, there is more control on resources: the scheduler has knowledge of the system by monitoring of the resource state, and therefore it is easier to obtain efficient schedulers. As this type of scheduling suffers from limited scalability and is not appropriate for large-scale Grids.

- **Decentralized Scheduling**

In decentralized or distributed scheduling there is no central entity controlling the resources. The autonomous Grid sites make it more challenging to obtain efficient schedulers. In decentralized schedulers, **the local schedulers play an important role**. The scheduling requests, either by local users or other Grid schedulers, are sent to local schedulers, which manage and maintain the state of the job queue. This type of scheduling is more realistic for real Grid systems of large scale, although decentralized schedulers could be less efficient than centralized schedulers.

- **Co-operative Scheduling**

In cooperative scheduling, each grid scheduler has the responsibility to carry out its own portion of the scheduling task, but all schedulers are working towards common system-wide reach. A feasible schedule is computed through the cooperation of procedures, rules, and Grid users.

- **Non–cooperative Scheduling**

In non cooperative cases, individual schedulers act alone as autonomous entities and arrive at decisions regarding their own optimum objects independent of the effects of the decision on the rest of system.

Scheduling policies

- **Adaptive Scheduling**

The changeability over time of the Grid computing environment requires adaptive scheduling techniques, which will take into account both the current status of the resources and predictions for their future status with the aim of detecting and avoiding performance deterioration. Rescheduling can also be seen as a form of **adaptive scheduling** in which running jobs are migrated to more suitable resources.

- **Hierarchical Scheduling**

In hierarchical scheduling, the computation is at **three levels**. **The top level is called the meta-level**, where we have a **super manager to control group merging/partitioning**. At this level, tasks are not scheduled directly, but reconfiguring the scheduler according to the characteristics of the input tasks. Such a process is called **meta-scheduling**. **The mediate level is called the group level**, where the manager in each group collaborates with each other and allocates tasks for the workers in the group. The purpose of collaborating between managers is to improve the load balance across the groups. Specifically, the managers ensure that the workload in the local ready lists is roughly equal for all groups. **The bottom level is called the within-group level**, where the workers in each group perform self-scheduling. The hierarchical scheduler behaves between centralized and distributed schedulers, so that it can adapt to the input task graph.

- **List Scheduling**

A list scheduling **heuristic prioritizes workflow tasks and schedules them based on their priorities**. The basic idea of list scheduling is to make an ordered list of processes by assigning them some priorities, and then repeatedly execute the following two steps until a valid schedule is obtained :1) Select from the list, the process with the highest priority for scheduling. 2) Select a resource to accommodate this process. The Priorities are determined statically before scheduling process begins. The first step chooses the process with the highest priority; the second step selects the best possible resource. Some of the list scheduling strategies are Highest level first algorithm , Longest path algorithm, Longest processing time, Critical path method etc.

- **Master Slave Scheduling**

In Master Slave Scheduling model, each job has to be processed sequentially **in three stages**. **In the first stage, the preprocessing task runs on a master machine**; in **the second stage, the slave task runs on a dedicated slave machine**; and in **the last stage, the post processing task again runs on a master machine**, possibly different from the master machine in the first stage.

Traditional Scheduling Algorithms

- **First Come First Serve:** the First Come First Serve (FCFS) for parallel processing and is aiming at the resource with the smallest waiting queue time and is selected for the incoming task. This can be called as **Opportunistic Load Balancing (OLB)** [12] or myopic algorithm.
- **Min_Min:** the Min_Min heuristic begins with the set of all unmapped tasks having the set of minimum completion times. Then task with the **overall minimum completion time** from the set is selected and assigned to the corresponding machine.
- **Minimum Completion Time:** the Minimum Completion Time (MCT) assigns each task, in arbitrary order, to the machine with the minimum expected completion time for that task [13]. This causes some tasks to be assigned to machines that do not have the minimum execution time for them.
- **Backfilling:** Backfilling is a scheduling optimization that allows a scheduler to make better use of available resources by running jobs out of order. The scheduler prioritizes the jobs in the queue according to a number of factors and then orders the jobs into a highest priority first (or priority FIFO) sorted list. Backfilling allows small jobs to initiate before larger queued jobs as the larger queued jobs resources currently unavailable.
- **Round Robin Technique:** It focuses on the fairness. RR uses the ring as its queue to store jobs. Each job in a queue has the same execution time and it will be executed in turn.
- **Earliest Deadline First:** Earliest Deadline First (EDF) or Least Time to Go is a dynamic scheduling algorithm used in real-time operating systems. It places processes in a priority queue. Whenever a scheduling event occurs (task finishes, new task released, etc.) the queue will be searched for the process closest to its deadline, the found process will be the next to be scheduled for execution.

Traditional Scheduling Algorithms

- **Minimum Execution Time:** in contrast to OLB, Minimum Execution Time (MET) assigns each task, in arbitrary order, to the machine with the best expected execution time for that task, regardless of that machine's availability.
- **Max Min:** the Max_Min heuristic is very similar to Min_min. The Max_min heuristic also begins with the set of all unmapped tasks. Then, the set of minimum completion times is found. Now, the task with the overall **maximum completion time** from the set is selected and assigned to the corresponding machine (hence said to be Max_Min).

Optimization Techniques

- Mathematical Programming
- Network Analysis
- Branch & Bound
- Genetic Algorithm
- Simulated Annealing Algorithm
- Tabu Search
- …

Iterative Improvement 1

- General method to solve combinatorial optimization problems

Principles:

1. Start with initial configuration
2. Repeatedly search neighborhood and select a neighbor as candidate
3. Evaluate some cost function (or fitness function) and accept candidate if "better"; if not, select another neighbor
4. Stop if quality is sufficiently high, if no improvement can be found or after some fixed time

Iterative Improvement 2

Needed are:

1. A method to generate initial configuration
2. A transition or generation function to find a neighbor as next candidate
3. A cost function
4. An Evaluation Criterion
5. A Stop Criterion

Iterative Improvement 3

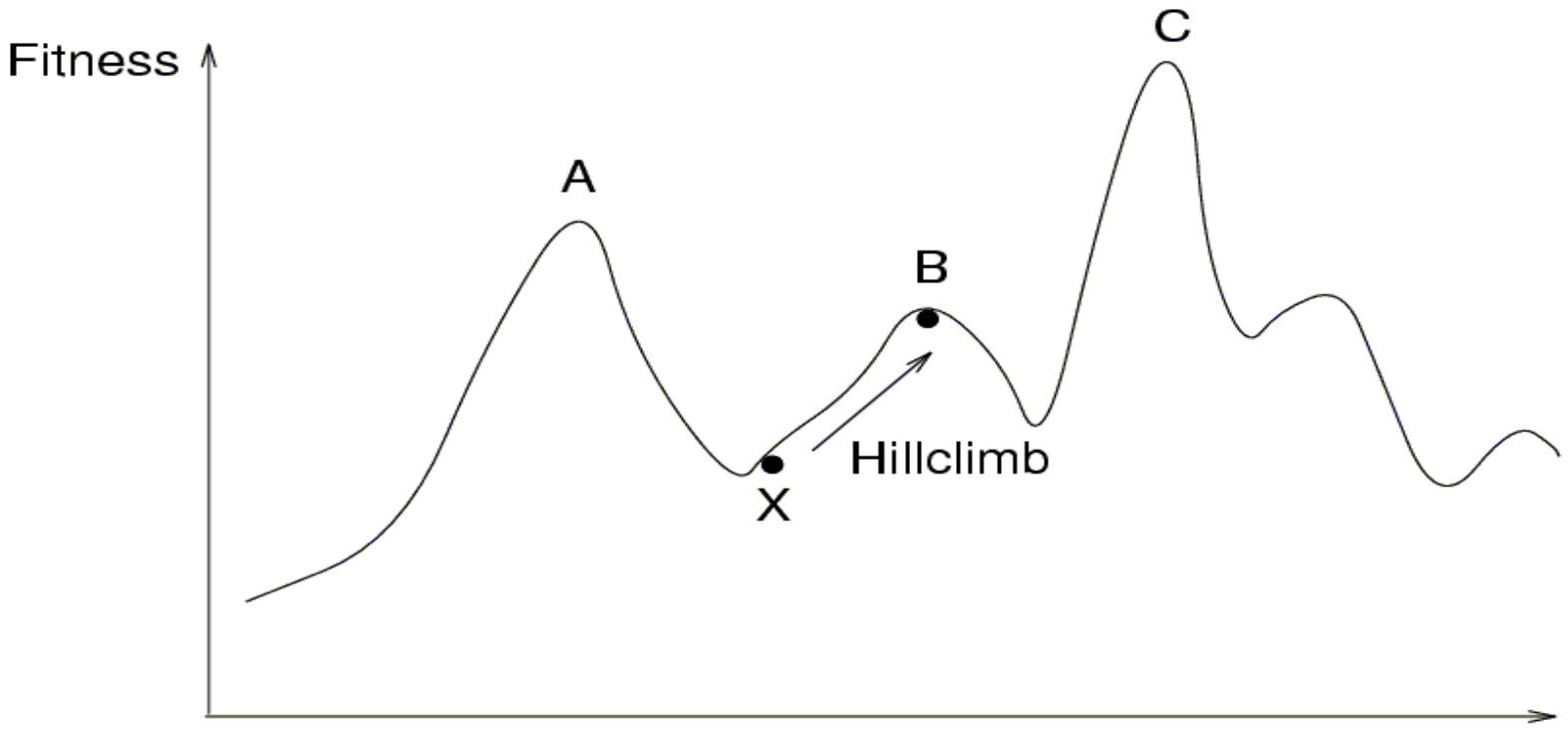
Simple Iterative Improvement or Hill Climbing:

- Candidate is always and only accepted if cost is lower (or fitness is higher) than current configuration
- Stop when no neighbor with lower cost (higher fitness) can be found

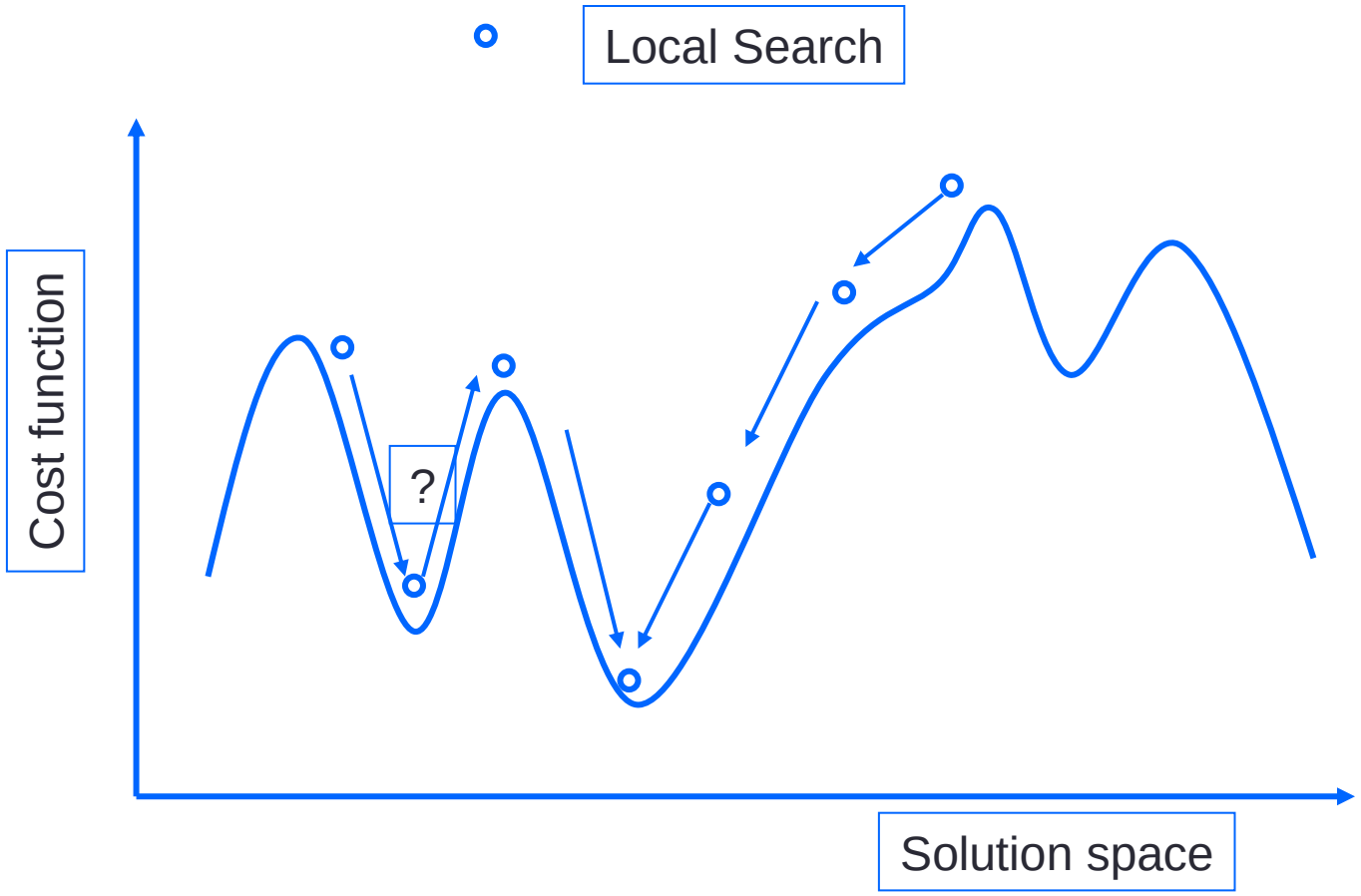
Disadvantages:

- Local optimum as best result
- Local optimum depends on initial configuration
- Generally, no upper bound can be established on the number of iterations

Hill climbing



Simulated Annealing



How to cope with disadvantages

1. **Repeat** algorithm many times with **different initial** configurations
2. Use information **gathered in previous runs**
3. Use a more complex Generation Function to jump out of local optimum
4. Use a more complex **Evaluation Criterion** that accepts sometimes (randomly) also solutions away from the (local) optimum

Improvement-based Scheduling Algorithms

- **Ant Colony Optimization:** A good scheduler should accommodate its scheduling policy according to the dynamicity of the entire environment and the types (nature) of jobs. Ant Colony Optimization (ACO) is an appropriate algorithm, which are dynamic in nature. ACO is a heuristic algorithm with efficient local search for combinatorial problems. ACO imitates the behavior of real ant colonies in nature to search for food and to connect to each other by pheromone laid on **paths traveled**. ACO has been used to solve NP-hard problems such as traveling salesman problem, graph coloring problem, vehicle routing problem, and so on.

Improvement-based Scheduling Algorithms

- **Genetic Algorithm:** It is an evolutionary technique for large space search. The general procedure of GA search and is as follows:

1) **Population generation:** A population is a set of chromosomes, representing a possible solution, can be a mapping sequence between tasks and machines. The initial population can be generated by other heuristic algorithms, such as Min_Min.
2) **Chromosome evaluation:** Each chromosome is associated with a **fitness value**, which is the make-span of the task machine mapping this chromosome represents. The goal of GA search is to find the chromosome with optimal fitness value.
3) **Crossover and Mutation operation:** Crossover operation selects a random pair of chromosomes and chooses a random point in the first chromosome, from that point to the end of each chromosome, crossover exchanges machine assignments between corresponding tasks. Mutation randomly selects a chromosome, then randomly selects a task within the chromosome, and randomly reassigns it to a new machine.
4) **Evaluation of the modified chromosome:** The chromosomes from this modified population are evaluated again.

The above defined steps complete **an iteration of the GA**. The GA stops when a predefined number of evolutions have been reached or all chromosomes converge to the same mapping. GA randomly selects chromosomes. More precisely, **Crossover** is the process of swapping certain sub-sequences in the selected chromosomes. **Mutation** is the random process of replacing certain sub-sequences with some task-mapping choices that are new to the current population. After **crossover and mutation**, a new **population** is generated. Then this new population is evaluated, and the process starts over again until some **stopping criteria** are met.

The stopping criteria can be, for example:

1) No improvement in recent evaluations;
2) All chromosomes converge to the same mapping;
3) A cost bound is met. GA is the most attracted and popular Nature's Law heuristic algorithm used in optimization problems.

Improvement-based Scheduling Algorithms

Simulated Annealing (SA): is a search technique based on the actual process of annealing.

- It is the thermal process of obtaining low-energy crystalline states of a solid. Very first level of the processing starts with melting the solid, the temperature is increased to melt the solid. The temperature is slowly decreased; particles of the melted solid arrange themselves locally, in a **stable "ground" state of a solid**.
- SA theory states that if temperature is lowered sufficiently slowly, the solid will reach thermal equilibrium, which is an optimal state. The thermal equilibrium is an **optimal task machine mapping (optimization)**, the temperature is the **total completion time of a mapping (cost function)**, and **the change of temperature is the process of dynamicity of mapping**.
- If the next temperature is higher, which means a worse mapping, the next state is accepted with certain probability. This is because the acceptance of some "worse" states provides a way to break out local optimality which occurs often in local search.

Improvement-based Scheduling Algorithms

- **Particle Swarm Algorithm:** Particle Swarm Optimization (PSO) is a proposed Algorithm motivated by social behavior of organisms such as **bird flocking** and **fish schooling**.
- PSO algorithm is not only a tool for optimization, but also a tool for representing socio cognition of human and artificial agents, based on principles of social psychology.
- PSO as an optimization tool provides a population-based search procedure in which individuals called particles change their position (state) with time.
- In a PSO system, particles fly around in a multi-dimensional search space. During this fly, each particle adjusts its position according to its own experience, and according to the experience of a neighboring particle, making use of the best position encountered by itself and its neighbor.
- PSO system combines local search methods with global search methods, attempting **to balance exploration and exploitation**.

Improvement-based Scheduling Algorithms

- **Game Theory:** The Grid scheduling problem is modeled using Game Theory (GT), which is a technique commonly used to solve economic problems.
- Each task is modeled as a player whose available strategies are the resources on which the task could run. The payoff for a player is defined to be the sum of the benefit value for running the task and all communication arriving at that task.
- GT, a job scheduling game is a game that models a scenario in which multiple selfish users wish to utilize multiple processing machines. Each user has a single job, and the user needs to choose a single machine to process it. The incentive of each user is to have his job run as fast as possible.
- Job scheduling games are the following set of problems:
  - given a set of machines and a set of jobs.
  - Each job is associated with a vector, corresponding to its size on each machine (i.e., is the processing time of job on a machine).
  - Players correspond to jobs. The strategy set of each player is the set of machines.
  - Given a strategy for each player, the total load on each machine is the sum of processing times of the jobs that chose that machine. Usually each player seeks to minimize the total load on its chosen machine.
  - The standard objective function is minimizing the total load on the most loaded machine (makespan minimization).

Improvement-based Scheduling Algorithms

- **Tabu Search (TS):** is about **short hop procedure to find the nearest local minimum solution within the solution space**.
- TS is a solution space search that keeps track of the regions of the solution space, which have already been searched so as not to repeat a search near these areas.
- TS is a meta-strategy for guiding known heuristics to overcome local optimality and has now become an established optimization approach that is rapidly spreading to many new fields.
- The method can be viewed as an iterative technique which explores a set of problem solutions, by repeatedly making moves from one solution to another solution located in the neighborhoods.
- These moves are performed with the aim of efficiently reaching an optimal solution by minimizing some objective functions.

Improvement-based Scheduling Algorithms

- **Fuzzy Algorithms:** Fuzzy Logic (FL) techniques to design an adaptive FL scheduler, which utilizes the FL control technology to select the most suitable computing node in the Grid environment.
- A fuzzy set is a set containing elements that have varying degrees of membership. There is fuzziness in all preemptive scheduling algorithms.
- A Fuzzy Neural Network was also proposed, to develop a high performance scheduling algorithm. The algorithms uses FL techniques to evaluate the Grid system load information, and adopt the Neural Networks (NN) to automatically tune the membership functions.
- Artificial Neural Network (ANN) is data-driven modeling tool that is able to capture and represent complex and non-linear input/output relationships.
- They are recognized as powerful and general technique for Machine Learning because of their nonlinear modeling abilities and robustness in handling noise ridden data.

EVALUATION OF GANG SCHEDULING PERFORMANCE AND COST IN A CLOUD COMPUTING SYSTEM

Ioannis A. Moschakis and Helen D. Karatza,
Journal of Supercomputing, Springer, 59(2), pp. 975-992 (2012).

Presented by: Dr. Faramarz Safi
Islamic Azad University, Najafabad Branch,
Esfahan, Iran.

Cloud Scheduling – A Performance Study (1/15)

- The model utilizes the concept of VMs which act as the computational units of the system.
- Initially the system includes no VMs but, depending on the computational needs of the jobs being serviced new VMs can be leased and later released dynamically.

Cloud Scheduling – The Simulation Model (2/15)

- The simulation model consists of a single cluster of VMs connected with a *Dispatcher VM (DVM)*.
- Initially the system leases no VMs so the cluster is empty.
- Depending on the workload at any specific moment the system has the ability to lease new VMs up to a total number of *Pmax* = 120.

Cloud Scheduling – The Simulation Model (3/15)

- Each VM incorporates its own task waiting queue, where the tasks of parallel jobs are dispatched by the DVM.

- The DVM also includes a waiting queue for jobs that were unable to be dispatched at the moment of their arrival due to either *inadequacy of VMs* or due to *overloaded VMs*.

Cloud Scheduling – The System Model (5/15)

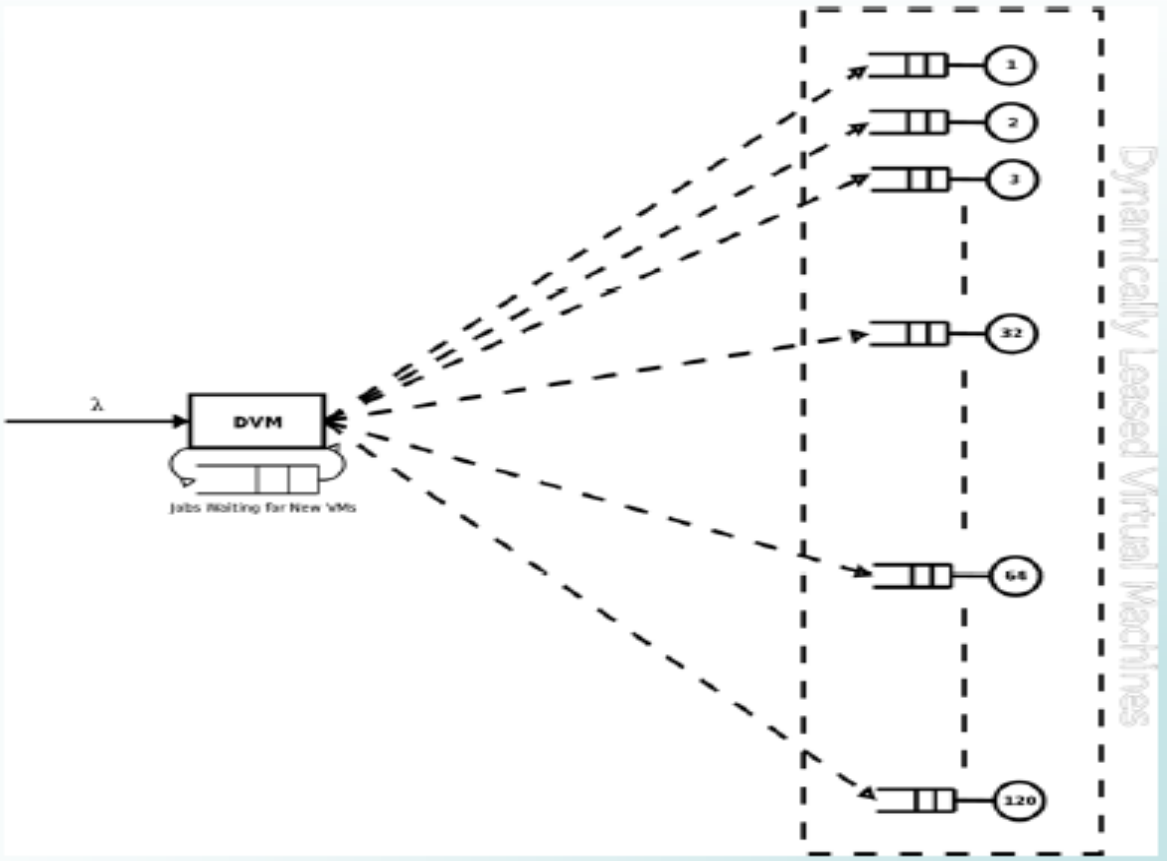


Fig. 16. The cloud queueing model

Cloud Scheduling – Jobs Characteristics (4/15)

- Jobs fall into two different categories of size:
  - **Lowly Parallel Jobs**, that have job sizes in the range [1...16] with a probability of q.
  - **Highly Parallel Jobs**, that have job sizes in the range [17..32] with a probability of $1-q$.
- The **job size coefficient *q*** determines what percentage of jobs belong to the first category.
- The system was examined under a q of 0.25, 0.50 and 0.75.

Cloud Scheduling - Dispatching (6/15)

- **Job Routing**

- The job entry point for the system is the DVM. If the degree of parallelism of a job is less than or equal to the number of the available VMs, the job is immediately dispatched.
- The allocation of VMs to tasks is handled by the DVM which employs the *Shortest Queue First (SQF).*
- Tasks that belong to the same job cannot occupy the same queue since gang scheduling requires that there exists a one-to-one mapping of tasks to server VMs.

Cloud Scheduling – Policies (7/15)

- Adaptive First Come Fist Served (AFCFS)
- Largest Job First Served (LJFS).

Cloud Scheduling - VM Handling (8/15)

VMs Lease / Release

- The *Cloud* provides users with the ability to quickly upscale or sub-scale their available resources.

- The addition of more VMs is accomplished through a virtual machine cloning process which involves the replication of a single initial state that all new virtual machines share.

- In this system model a delay is introduced which refers to the time that the VM cloning process will take to create a stated number of new VMs.

Cloud Scheduling - Inadequate VMs (9/15)

- The ***lease/release* cycle** of VMs happens dynamically while the system is in operation.

- **Inadequate VMs.** When a large job arrives and the system has an inadequate amount of VMs to serve the job, then the newly arrived job enters the waiting queue of the DVM and waits while the system provisions for new virtual machines.

- This procedure obviously involves a certain delay that refers to the real world delay of cloning a virtual machine and inserting it in the VM cluster.

Cloud Scheduling - Overloaded VMs (10/15)

- **Overloaded VMs.** When a new job arrives, the system checks the *Average Load Factor (ALF)* of the available VMs:

$$ALF = \frac{\sum_{i=1}^{P_l} t_i}{P_l}$$

- where t_i is the number of tasks currently assigned to $\mathbf{VM_i}$ and P_l is the number of VMs leased by the system at that moment.

- The *ALF* threshold is set to 10 tasks per VM. Should *ALF* surpass this threshold, the system provisions for new VMs equal to the degree of parallelism of the arriving job.

Cloud Scheduling - Releasing VMs (11/15)

- Server VMs can also be released when they are not needed.
- Idle servers in the Cloud are costly for the user. The process of releasing a VM involves a certain delay.

- VMs are released only if certain criteria are met:
  - The VM is currently idle and the VM's task waiting queue is also empty.
  - The removal of the VM from the system will not cause a new shortage of VMs, for jobs that are waiting in the DVM's queue for new VMs to be leased.

Cloud Scheduling – Performance / Cost (12/15)

- The use of the Cloud is "**cost- associative**":
- One pays only for the computing time which is equivalent to the total lease time of virtual machines.

- *Cost to performance efficiency* view.

- *Total lease time* (LT) of virtual machines while the system is in operation:

$$LT = \sum_{i=1}^{P_{tot}} T_{lease(i)}$$

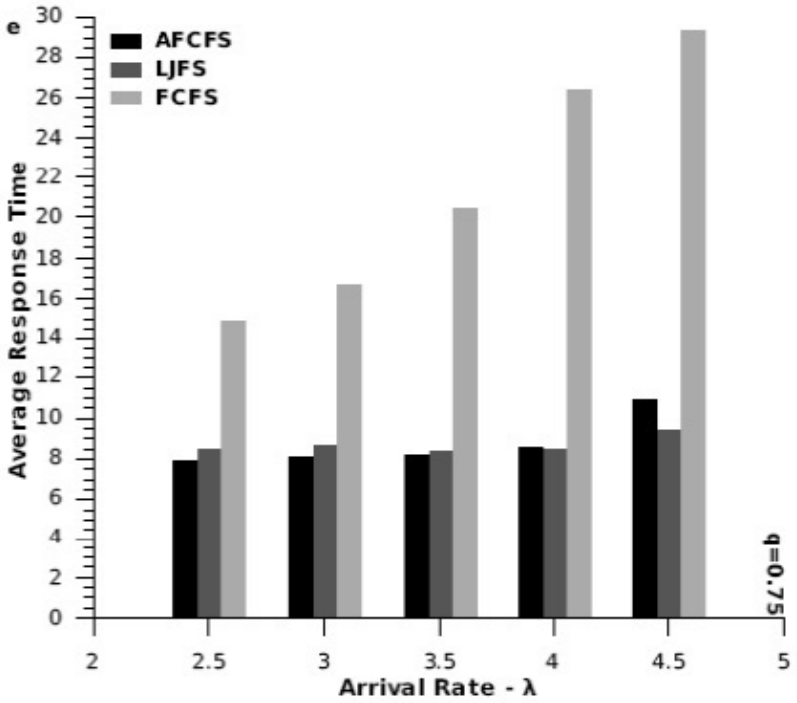
Cloud Computing – Parameters (13/15)

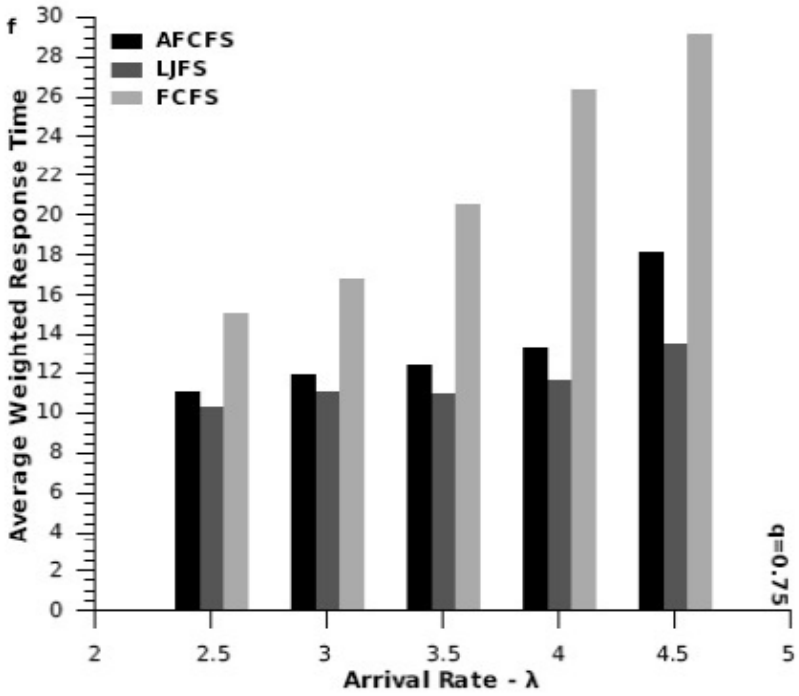
- In order to compare LJFS and AFCFS **cost-performance wise**, the following metrics are used:

  - Average and Weighted Response and Waiting Time in conjunction with Slowdown metrics.

  - A metric termed **Cost-Performance Efficiency** (CPE) was devised to evaluate the gain in response time in relation to the cost.

Cloud Scheduling – Simulation Results (14/15)





Cloud Scheduling – Simulation Results (15/15)

Table 2 Cost-to-Performance Efficiency AFCFS-LJFS

λ	$q = 0.25$	$q = 0.5$	$q = 0.75$
1.75	−10.1232	-	-
2	1.3171	-	-
2.25	4.2493	6.3589	-
2.5	14.8915	4.2130	0.1413
2.75	17.3365	2.2873	-
3	-	8.1537	0.4645
3.25	-	21.5644	-
3.5	-	-	4.7581
4	-	-	8.4878
4.5	-	-	22.9386