

End Semester Project

*Operating System*



*December 26, 2016*

***group memebers***

*Ali Raza Chishti Rohma Shakeel Muhammad Gulzain Ali Khan*

# **Paper Details**

## ***Research Paper Title***

*Adaptive work-stealing with parallelism feedback*

## ***Authors***

* *Kunal Agrawal*
* *Charles E. Leiserson*
* *Yuxiong He*
* *Wen Jing Hsu*

## ***Journal Issue Name***

* *Volume 26 Issue 3,*
* *September 2008*
* *Article No. 7*

## ***Paper Description***

Scheduling of multi-processors in a shared multiprogramming systems can be re-structured as two-level scheduling as

* A kernel-level job scheduler (for allocation of CPU).
* A user-level thread scheduler (for scheduling of user tasks).

In this manner, this paper shows a randomized work-stealing thread scheduler using fork-join commands as we have studied in class for Operating systems. It also provides continual parallelism feedback to the job scheduler in the form of requests for CPU. This research paper model the job scheduler as the thread scheduler's adversary, challenging the thread scheduler to be robust to the operating environment as well as to the job scheduler's administrative policies. To analyze the performance of this adaptive thread scheduler, they introduced a new technique as “trim analysis”, which allows to prove that the thread scheduler performs poorly on no more than a small number of time steps, exhibiting near-optimal behavior on the vast majority.

# **Questions to be Answered**

## ***Why did you select this paper?***

We select this paper due to the following reasons:

* We have studied multi-threading and its handling by CPU on just one processor in the class but it also gives us information of process handling at multi-cores as well.
* We use an algorithm named A-Steal algorithm for parallel serving of many jobs with its feedback.
* It shows the allocation of CPU scheduler.
* It provides us information related to Task scheduler handling at user side.
* It helps us to demonstrate the work-stealing environment for CPU for the completion of tasks.

## ***How is this paper related to the OS course?***

This paper is related to OS course because it covers a lot that part of scheduling and multi-processing at kernel level so it provides us better knowledge about multi-processing / multi-threading using multi-processors to get parallelism so that it will help us to improve our OS knowledge on the existing computers of multi-cores. It helps us to enhance our knowledge on these topics

1. CPU multi-tasking
2. Scheduling at kernel level.
3. Scheduling at user level.
4. Division of task in multi-threads using for-join commands.
5. We can analyze our adaptive system’s performance using “**trim** **analysis**” which proves that if our task will be in small steps then thread schedule will perform poorly.

## ***What is the main problem that this paper is attempting to solve?***

In this paper, following problems are to be solved:

* Multi-tasking using multi-cores so that allocation of resources can be maximized.
* User will be able to do scheduling on his side so that he may be able to run the processes according to his will.
* We will be able to test this adaptive system in so many adverse environments so that we can analyze its performance in different conditions and will be able to generate reports on it.

## ***What are the main contribution(s) of this paper? (if it is a review paper, what are the main approaches that it covers?)***

It divides the multi-processor scheduling in 2 different environments, where a kernel-level job scheduler allots processors to jobs and a user-level thread scheduler schedules the work of a job on its allotted processors.

It uses A-STEAL algorithm for large parallel servers where many jobs share a common multiprocessor resource and in which the number of processors available to a particular job may vary during the job’s execution. For example the job scheduler never allots a job more processors than requested by the job’s thread scheduler, this algorithm guarantees that the job completes in near-optimal time while utilizing at least a constant fraction of the allotted processors

It also demonstrates the benefits of adaptive system and provides feedback through “Trim Analysis”

It shows the measurement of the performance of A-STEAL on a simulated multiprocessor system using synthetic workloads. For jobs with sufficient parallelism, its experiments confirm that A-STEAL provides almost perfect linear speedup across a variety of processor availability profiles. It also shows comparison between A-STEAL and the ABP algorithm, an adaptive work-stealing thread scheduler developed by **Arora** which does not employ parallelism feedback. On moderately to heavily loaded machines with large numbers of processors, A-STEAL typically completed jobs more than twice as quickly as ABP.

## ***Give a real world example where the method(s)/approach(es) devised/reviewed in this paper can be effectively used.***

We can provide its example as:

Teachers often ignore the lowest score while computing a student’s grade. In the same way, in the Olympic Games, the lowest and the highest scores are sometimes ignored when computing an athlete’s average. In theoretical computer science, when an adversary is too powerful, we sometimes make statistical assumptions about the input to render the analysis tractable, but statistical assumptions may not be valid in practice.

*Trim* *analysis* may prove itself of value for analyzing such problems. ***A-STEAL***, as presented, uses full information about the previous quantum to estimate the desire of the current quantum. Collecting perfect information might become difficult as the we have multiple processors, especially when the number of processors exceeds the length of quantum. A-STEAL only estimates the desire, however, approximate information should be enough to provide feedback.