# INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

## Department of Computer Science and Engineering

CS528 (**High Performance Computing**) End Semester Examination

**Date: 22th Nov 2016 Timing: 3 Hrs Full Marks: 50 Answer all questions**

## Q1 [12 Marks] [Topic: Basic Scheduling]

A. [6 marks] Design an efficient scheduling algorithm to minimize the overall execution time $C_{max}$ for execution of DAG of tasks with arbitrary execution time on unbounded number of processors. ($P_\infty|prec|C_{max}$)

***Ans: As soon as possible scheduling, whenever a task is ready (all the parents of the task have executed), schedule the task to execute the task. Start scheduling task from top and schedule one by one.***

B. [6 marks] Given $n$ real time tasks, where each task is characterized by arrival time $a_i$, execution time $e_i$ and deadline $d_i$. These tasks need to be run on unbounded number of homogenous processor with power consumption model for processor as $P = S + k.f^3$, where $S$ is static power consumption and $f$ is frequency of operation of the processor, $S > 2.k$ and $0 < f \leq 1$. Execution time of a task at frequency $f_j$ is $e_{ij}=e_i/f_j$. Design an efficient algorithm to minimize the overall energy consumption of the system without missing deadline of any tasks.

*Ans: Energy consumption of processor running a task at frequency $f_j$ is $E_{ij}= e_i/f_j*(S + k.f_j^3)=e_i(S/f_j+k f_j^2)$. So minimum energy will be at $f_c=(S/2.k)^{1/3}$. As S>2k, so $f_c>1$. If we assume, there is no setup/boot up time, run all the task at f=1, will give minimum energy.*

*When a task arrives to the system, fetch a processor and run the task with f=1.*

## Q2 [14 Marks] [Topic: Data Center and Cloud Environment]

A. [6 marks] Cloud system load prediction uses exponentially weighted moving average (EWMA) model to predict the incoming load to the system. $E(t) = \alpha E(t\text{-}1)+(1\text{-}\alpha) O(t)$, $0 \leq \alpha \leq 1$, where $E(t)$ is predicted and $O(t)$ is observed load at time $t$ and $\alpha$ is trade of between stability and responsiveness. Assume $0 \leq O(t) \leq R$ for all t.

**a.** Mention one scenario, where this model gives the **worst** result and calculate the amount of error in prediction if you can calculate.

*Ans: Load can be from 0 to R, suppose you predict E for next time slot, but actual load comes with higher error which is if E is bellow R/2 (same as (R-E)>E) then O=R else E=0.0, based on that system always tries to adjust the E values to (E+O)/2 for the next. And same situation repeats then Average error will be above R/2 (which is R-E if E is less than R/2 and Error is E-0.0 if E>R/2). As every time readjust E to (E+O)/2 with alpha=0.5. The error will be around 66%.*

*But every time fix E to be R/2, then error will be 50% and if we say E=O (**α**=0) of previous then it will go for 100% error. To increase the error, the load is chosen to be 0 or R which agai predicable using other model. So another funny situation, a guy is sending a load exactly opposite to what you predict (he know your predicting algorithm and try to create problem on your prediction), If you know this fact, simply changing the prediction model (**α=1**), will predict with 0% error.*

*If we assume all the loads are coming randomly, error will be **α=1 and E=50, a minimum of 25% and α=0 (E=O) with maximum of 33.3%.***

**b.** Mention one scenario for the **best** result and calculate the amount of error in prediction.

*Ans: When all the observed values are constant and take only one value, load is same over time. There will not be any error in prediction.*

**B.** [4 marks] Given a data center with virtually infinite homogenous servers. If a server runs at utilization less than 40% then there is no requirement of pushing cool air in to the server. Power consumption of one sever can be calculated as ***P= S+k.u***, where ***S*** is static power consumption, ***k*** is a constant and ***u*** is utilization of the serve and static power consumption is negligible as compared to dynamic part, ***S<10.k***. Power consumption in producing cool air for a server is two time costly as compared to computing, which is ***$P_{ac}$=2*k***. Design an efficient way to schedule online real time tasks (***$a_i$, $d_i$, $e_i$***) to minimize the energy consumption of the system.

*Ans: Assume one task can run on one server at any instant of time (one core per server). Let assume S=9.999=10, k is 100. Pac=200, running server at above 40% utilization cost minimum of 10+100*0.4+200=250. And running below 40% will cost up to maximum 10+100+0.4=140. So it is advisable to run all the server bellow this utilization whenever possible and merge many tasks to put with in the allowable range (40% utilization). Suppose a task require utilization above 40% when $e_i/(d_i-a_i)>0.4$, run that task separately on one server and when it finish switch of that server.*

*For other tasks, try to merge many tasks to make utilization up to 40% to reduce the static part. We can use any greedy/heuristic approach to do the same.*

C. [4 marks] A ***low cost air line announced a festive offer season*** and over book ***10%*** of his seat by assuming ***10%*** of customer will not turn up at the time of flight. Cost of ticket is ***C*** per person and in actual scenario ***92%*** chances that a customer reported to airline at the time flying. If the flight is full then the airline need to do some alternate arrangement and which cost ***1.25*C*** per passenger to the airline. Calculate the profit by doing the overbooking.

D. *Ans : Lets assume flight have 100 seats, booked set 110. Number of people turn up at the time of flight 110*0.92=101.2. As the flight will go with fully capacity and 1.2 people need to be accommodated by alternate arrangement, so it will cost 1.2*1.25C=1.5C. Benefit of overbooking is 10C-1.5C=**8.5C***

**Q3 [12 Marks] [Topic: Auto tuning and benchmarking]**

A. [6 marks] A ***communication intensive benchmark*** uses ***N*** processes, run for ***I*** iterations and in each iteration it do the same type of work and communications between processes. For this application, each process needs to be mapped to one core. Overhead of collecting information about one-one effective distance between the allocated processor takes ***2s*** and running iteration takes time in hours. Formulate a method to ***remap the process ranking*** based on first iteration and use it in further iteration to reduce the overall communication cost and implied execution time.

*Ans: Calculate of pair wise distances among all processor can be done in 2s. Application is **communication intensive and** in each iteration it do the same type of work and communications between processes. Suppose by profiling communication, we can get pair wise communication amount for all process. Now we need to remap the process ranking such that overall communication cost is minimized and it may reduce the effective execution time. We have distance matrix Dist[N][N] and data communication matrix Comm[N][N]. Simply need to remap/rerank the process using a function R so that*

*Sum_ij Dist[i][j]*Comm[R[i]][R[j]] is minimized. This is similar to quadratic assignment problem. We generally use Simulated Annnealng to solve this problem as QAP is a hard problem.*

B. [6 marks] Given a HPC system with ***200*** nodes and each node having ***20*** cores. ***20*** MPI process can run smoothly on a node. Users are allowed to run their applications with required number of nodes (***$N_r$***) and required number of process (***$N_t$***) on each node. From scheduler point of view, propose some user constraints on ***$N_r$*** and ***$N_t$*,** so that scheduling become easier for the resource managers and minimize the users wait time in the queue for any request. Analyze the effect of ***your proposed constraints*** on scheduling.

*Ans:*

- *Suggest user not to specify Nr, they will say only Nt: Scheduler have easier way to manage, as user have requested for only number of processors, scheduler have freedom to give Nt number of cores to users from the free pool from any number of nodes.*
    - *Require to maintain free pool of cores, every cores need to be numbered from 0 to 4000. Using a binary array of 4000 size is enough for management. Here also, we may need to deal with fragmentation (but some time it is not necessary) if we try to allocate the requested number of cores by a user to be nearby cores.*
- *Suggest users to specify only multiple of 20. So that scheduler will simply allocate request number of node to user.*
    - *There may be some wastage if user requirement is not multiple of 20*
    - *For Scheduler, it is easy to manage 200 nodes, can be done with a binary array of size 200.*