D: Set of all devices
T: Set of all time steps
J: Set of all jobs
$R \subset J \times J \times \mathbb{Z}^+$: Set of dependencies $n \times a \to b$ (execute A n times for every B)
$Threads : D \to \mathbb{Z}^+$: Returns the number of threads on a device.
$ExecTime : J \times D \to \mathbb{Z}$: Returns the number of timesteps it takes to execute a task on a given device.
$EncodeTime : J \times D \times D \to \mathbb{Z}$: Returns the number of timesteps it takes to serialize the output of a particular job destined for a particular device.
$TransitTime : J \times D \times D \to \mathbb{Z}$: Returns the number of timesteps it takes to send the output of a particular job from one device to another after it has been serialized.

$q_{j,d,t} =$ number of job j that should be completed on device d by time t
$m_{j,d,d2,t}$ = the number of job j outputs that have moved from device d to device $d2$
$z \in T$ = final time

$$\forall_T t \forall_D d : \sum_j^J q_{j,d,t+ExecTime(j,d)} - q_{j,d,t} + \sum_j^J \sum_{dst}^D m_{j,d,dst,t+EncodeTime(j,d,dst)} - m_{j,d,dst,t} \leq Threads(d)$$

A computer can only perform as many operations at a time as it has threads.

$$\forall_T t \, \forall_J j \, \forall_D d : q_{j,d,t} \geq \sum_{dst}^D m_{j,d,dst,t+ExecTime(j,d)+EncodeTime(j,d,dst)+TransitTime(j,d,dst)}$$

A job has to have been completed and it's outputs serialized and transmitted before it can exist on another device.

$$\forall_T t \, \forall_D d \, \forall_R a, b, n : \sum_{src}^D n m_{a,src,d,t} \geq q_{b,d,t+ExecTime(b,d)}$$

A job would have needed it's inputs when it started executing.

$$\forall_J j : \sum_d^D q_{j,d,last(T)} \geq Count(j)$$

All jobs must be executed

$$\forall_T t \, \forall_D d \, \forall_J j : t(q_{j,d,t+1} - q_{j,d,t}) \leq z$$

The time of the last job must be less than some objective
(**Approximation**: undefined behavior at the end for multithreaded machines likely prevents the

model from performing simultaneous finishes, though this isn't exact behavior anyways.)
Objective Function:

$$= z$$

(TODO) Incremental Reformulation

We might be able to take larger time steps and reformulate the equations so that multiple jobs can finish within the same time step on a single thread. This would allow for more incremental recomputation