Mathematical Model (Across Time Intervals)

Sets

- $M = \{1\ (\text{t4g.nano}), 2\ (\text{t4g.medium}), 3\ (\text{t4g.xlarge}), 4\ (\text{r8g.large}), 5\ (\text{c8g.xlarge}), 6\ (\text{r8g.2xlarge}),$
  $7\ (\text{c8g.4xlarge}), 8\ (\text{c8g.8xlarge}), 9\ (\text{m8g.8xlarge}), 10\ (\text{r8g.8xlarge}), 11\ (\text{m8g.12xlarge}), 12\ (\text{c8g.16xlarge})\}$
  = Set of Machine Types
- $T = \{1\ (\text{12am--3am}),\ 2\ (\text{3am--6am}),\ 3\ (\text{6am--9am}),\ 4\ (\text{9am--12pm}),\ 5\ (\text{12pm--3pm}),$
  $6\ (\text{3pm--6pm}),\ 7\ (\text{6pm--9pm}),\ 8\ (\text{9pm--12am})\}$
  = Set of Time Intervals
- S_t = Set of given tasks arriving in interval $t \in T$
- $J_t = \{1, 2, \ldots, |S_t|\}$ = Index set of potential instances per machine type in interval $t \in T$

Decision Variables

- x_{ijkt} : 1 if task $k \in S_t$ is assigned to instance $j \in J_t$ of machine type $i \in M$ during interval $t \in T$, 0 otherwise
- y_{ijt} : 1 if instance $j \in J_t$ of machine type $i \in M$ is used in interval $t \in T$, 0 otherwise
- f_{it} : number of instances of machine type $i \in M$ active at the end of interval $t \in T$
- g_{it} : number of new instances of machine type $i \in M$ spun up in interval $t \in T$
- r_{it} : number of active instances of machine type $i \in M$ shut down at beginning of interval $t \in T$

Parameters

- c_i : cost per hour in $ofmachinetype$i \in M$
- p_i : CPU limit in vCPUs of machine type $i \in M$
- q_i : Memory limit in GiB of machine type $i \in M$
- m_k : CPU requirement in vCPUs of task $k \in S_t$
- n_k : Memory requirement in GiB of task $k \in S_t$
- W : size of time window in hrs for which tasks are executing
- $\alpha = 0.1$: instance startup time in hrs (6 min)
- $s_i = \alpha c_i$: startup cost in Dollars of an instance of machine type $i \in M$

Objective Function

$$\min \sum_{t \in T} \sum_{i \in M} (W c_i f_{it} + s_i g_{it})$$

Constraints

- $\sum_{k \in S_t} m_k x_{ijkt} \leq p_i\,, \quad \forall i \in M, j \in J_t, t \in T$ `(CPU constraint per instance per machine type)`
- $\sum_{k \in S_t} n_k x_{ijkt} \leq q_i\,, \quad \forall i \in M, j \in J_t, t \in T$ `(Memory constraint per instance per machine type)`
- $\sum_{i \in M} \sum_{j \in J_t} x_{ijkt} = 1\,, \quad \forall k \in S_t, t \in T$ `(each task can only be assigned to 1 specific instance)`
- $y_{ijt} \leq \sum_{k \in S_t} x_{ijkt} \leq |S_t|\, y_{ijt}\,, \quad \forall i \in M, j \in J_t, t \in T$ `(link x and y constraint)`
- $\sum_{j \in J_t} y_{ijt} = f_{it}\,, \quad \forall i \in M, t \in T$ `(active instance count relationship/constraint)`
- $f_{it} = f_{i(t-1)} + g_{it} - r_{it}\,, \quad \forall i \in M, t \in T$ `(flow balance constraint)`
- $r_{it} \leq f_{i(t-1)}\,, \quad \forall i \in M, t \in T$ `(shut down cannot exceed previous active)`
- $x_{ijkt} \in \{0, 1\}\,, \quad \forall i \in M, j \in J_t, k \in S_t, t \in T$
- $y_{ijt} \in \{0, 1\}\,, \quad \forall i \in M, j \in J_t, t \in T$
- $f_{it}, g_{it}, r_{it} \geq 0\,, \quad \forall i \in M, t \in T$
- $f_{i0} = 0\,, \quad \forall i \in M$

Solution (Across Time Intervals)

In [2]:
```
function check_optimality(m)
    stat = termination_status(m)
    if stat != MOI.OPTIMAL
        println("Solver did not find an optimal solution: $stat")
    end
end;
```

In [7]:
```
using CSV, DataFrames
```

```julia
# Read EC2 types
ec2_df = CSV.read("ec2_subset.csv", DataFrame)

# Read tasks (one CSV per time interval)
tasks_df = Dict()
for t in 1:8
    tasks_df[t] = CSV.read("tasks_t$(t).csv", DataFrame)
end

M = collect(1:nrow(ec2_df)) # machine types
println("Number of machine types: ", length(M))
machine_name = Dict(i => ec2_df.Type[i] for i in M)

c = Dict(i => ec2_df.Cost[i] for i in M) # cost per hour of machine type
p = Dict(i => ec2_df.vCPUs[i] for i in M) # CPU limit of machine type
q = Dict(i => ec2_df.Memory[i] for i in M) # Memory limit of machine type

W = 3 # time interval window size
α = 0.1 # instance startup time
s = Dict(i => α * c[i] for i in M) # startup cost in $ of machine type
T = collect(1:8) # time intervals

S = Dict() # Set of tasks in each time interval
J = Dict() # Set of instances in each time interval
m = Dict() # CPU req
n = Dict() # Mem req

for t in T
    df = tasks_df[t]
    S[t] = collect(1:nrow(df))  # Task indices
    J[t] = collect(1:nrow(df)) # Instance indices
    println("Number of tasks in time interval $t: ", length(S[t]))
    m[t] = Dict(k => df.cpu_cores[k] for k in S[t])
    n[t] = Dict(k => df.mem_gb[k] for k in S[t])
end;
```

```
Number of machine types: 12
Number of tasks in time interval 1: 27
Number of tasks in time interval 2: 18
Number of tasks in time interval 3: 20
Number of tasks in time interval 4: 22
Number of tasks in time interval 5: 27
Number of tasks in time interval 6: 32
Number of tasks in time interval 7: 24
Number of tasks in time interval 8: 34
```

In [ ]:

```julia
using JuMP, HiGHS
model = Model(HiGHS.Optimizer)

@variable(model, f[i in M, t in T] >= 0) # number of active instances of type i at end of interval t
@variable(model, g[i in M, t in T] >= 0) # number of instances of type i started at beginning of interval t
@variable(model, r[i in M, t in T] >= 0) # number of instances of type i terminated at beginning of interval t

# time varying binary variables
x = Dict()
y = Dict()

for t in T
    x[t] = @variable(model, [i in M, j in J[t], k in S[t]], Bin) # task k is assigned to instance j of type i in interval t
    y[t] = @variable(model, [i in M, j in J[t]], Bin) # instance j of type i is active in interval t
end

@objective(model, Min, sum(W*c[i]*f[i,t] + s[i]*g[i,t] for i in M, t in T))

for t in T, i in M, j in J[t]
    @constraint(model, sum(m[t][k] * x[t][i,j,k] for k in S[t]) <= p[i]) # CPU constraint
    @constraint(model, sum(n[t][k] * x[t][i,j,k] for k in S[t]) <= q[i]) # Memory constraint
end

for t in T, k in S[t]
    @constraint(model, sum(x[t][i,j,k] for i in M, j in J[t]) == 1) # each task is assigned to one instance
end

for t in T, i in M, j in J[t]
    @constraint(model, y[t][i,j] <= sum(x[t][i,j,k] for k in S[t])) # instance is active if it has tasks assigned
    @constraint(model, sum(x[t][i,j,k] for k in S[t]) <= y[t][i,j] * length(S[t])) # instance can only have tasks assigned if
end

for t in T, i in M
    @constraint(model, sum(y[t][i,j] for j in J[t]) == f[i,t]) # number of active instances at end of interval t
end

for i in M
    @constraint(model, f[i,1] == g[i,1] - r[i,1])  # number of active instances at beginning of interval 1

    for t in 2:length(T)
        @constraint(model, f[i,t] == f[i,t-1] + g[i,t] - r[i,t]) # flow balance
        @constraint(model, r[i,t] <= f[i,t-1]) # # of instances terminated cannot exceed # of instances at beginning of interv
```

```
    end
end

set_silent(model)
optimize!(model)
check_optimality(model)
```

```
Set parameter Username
Set parameter LicenseID to value 2650823
Academic license - for non-commercial use only - expires 2026-04-13
Solver did not find an optimal solution: MEMORY_LIMIT
```

In [ ]:

```
println("Minimized total cost: ", objective_value(model))
for t in T
    println("Time interval $t:")
    for i in M
        println("  Machine type $(machine_name[i]):")
        println("    Active instances: ", value(f[i,t]))
        println("    Started instances: ", value(g[i,t]))
        println("    Terminated instances: ", value(r[i,t]))
        for j in J[t]
            if value(y[t][i,j]) > 0.5
                println("      Instance $j is active")
                for k in S[t]
                    if value(x[t][i,j,k]) > 0.5
                        println("        Task $k is assigned to instance $j")
                    end
                end
            end
        end
    end
end
```