D2: LP Formulation

October 26, 2020

Model: GPU Type based

- *Input Parameters:*

  $\mathbb{M}$ = Set of modules in a chain. A module is denoted as $m \in \mathbb{M}$

  $\mathbb{DAG}$ = A chain is represented as a DAG. A DAG is also called Precedence Graph, which can be represented as:

$$\mathbb{DAG}_{l,m} = \begin{cases} 1, & \text{if module } l \in \mathbb{M} \text{ directly precedes } m \in \mathbb{M} \\ 0, & \text{otherwise} \end{cases}$$

  $\mathbb{G}$ = Available GPU Types.

  Example: If $\mathbb{G} = \{\text{1080Ti, 2080Ti, V100}\}$

  P = List of offline profiles of each module on each GPU type.

  $P_{m,g}$ = List of profiles of module $m \in \mathbb{M}$ on a GPU type $g \in \mathbb{G}$.

$$P^k_{m,g} = \{< b^k_{m,g}, p^k_{m,g}, l^k_{m,g}, d^k_{m,g} >\},$$

  where $P^k_{m,g}$ denotes one <batch, parallel, latency, duration> configuration.

  C_g = List of cost of running a GPU type $g \in \mathbb{G}$.

  $S_{l,m}$ = Scaling factor from module $l \in \mathbb{M}$ to $m \in \mathbb{M}$. Namely, each input to module l will generate $S_{l,m}$ outputs, which will serve as inputs to module m.

  R_m = Input rate of module $m \in \mathbb{M}$

$$R_m = \sum_{\substack{l \in \mathbb{M} \\ \mathbb{DAG}_{l,m}=1}} R_l \times \mathrm{S}_{l,m}$$

$$R_{m0} = \Omega_{in}, \text{ where } \Omega_{in} \text{ is input rate to source module } m0$$

  L_{SLO} = Latency SLO

- *Decision Variables:*

  $X^k_{m,g}$: Allocation variable

$$X^k_{m,g} = \begin{cases} 1, & \text{if module } m \in \mathbb{M} \text{ is scheduled on } g \in \mathbb{G} \text{ with profile configuration } P^k_{m,g} \\ 0, & \text{otherwise} \end{cases}$$

  $R^k_{m,g}$: Input rate of a model m on GPU g with profile configuration $P^k_{m,g}$

$$R^k_{m,g} = \begin{cases} > 0, & \text{if } X^k_{m,g} = 1 \\ 0, & \text{otherwise} \end{cases}$$

$U^k_{m,g}$: Capacity allocated to a model m on GPU g with profile configuration $P^k_{m,g}$

$$U^k_{m,g} = \begin{cases} \frac{R^k_{m,g}}{\frac{b^k_{m,g} \times p^k_{m,g}}{d^k_{m,g}}}, & \text{if } X^k_{m,g} = 1 \\ 0, & \text{otherwise} \end{cases}$$

ST_m = Starting time for executing module m. This variable is required to calculate latency along critical path in the $\mathbb{DAG}$.

L_{max} = Critical latency denoted by sum of latency of modules on the critical path (longest path) of the DAG. Thus, critical latency is the max of latency along all paths of the DAG from source to sink module.

- *Objective Function:*

  Minimize cost of running all modules of a chain on GPU types $\mathbb{G}$.

$$\min \sum_{m \in \mathbb{M}} \sum_{g \in \mathbb{G}} C_g \times \sum_{k=0}^{|P_{m,g}|} U^k_{m,g}$$

- *Constraints:*

1. A module $m \in \mathbb{M}$ running on gpu type $g \in \mathbb{G}$ can take a unique offline profile configuration.

$$\sum_{m \in \mathbb{M}} \sum_{k=0}^{|P_{m,g}|} X^k_{m,g} \leq 1 \qquad \forall (g \in \mathbb{G})$$

2. Total input rate of module $m \in \mathbb{M}$ running on all the gpus should equal to R_m.

$$\sum_{g \in \mathbb{G}} \sum_{k=0}^{|P_{m,g}|} X^k_{m,g} \times R^k_{m,g} = R_m \qquad \forall (m \in \mathbb{M})$$

3. If module $l \in \mathbb{M}$ precedes $m \in \mathbb{M}$, the module m can process an input after module l has produced output for the same input.

$$ST_m \geq ST_l + X^k_{l,g} \times l^k_{l,g} \qquad \forall (l \in \mathbb{M}, m \in \mathbb{M}, 0 \leq k \leq |P_{l,g}|, g \in \mathbb{G}), \text{ where } \mathbb{DAG}_{l,m} = 1$$

4. Critical latency L_{max} denoted by sum of latency of modules on the critical path (longest path) of the DAG.

$$L_{max} \geq ST_m + X^k_{m,g} \times l^k_{m,g} \qquad \forall (l \in \mathbb{M}, m \in \mathbb{M}, 0 \leq k \leq |P_{m,g}|, m \in \mathbb{M}, g \in \mathbb{G})$$

5. Critical latency should be within latency SLO L_{SLO}

$$L_{max} \leq L_{\text{SLO}}$$

Model: Independent GPU Instances

- *Input Parameters:*

  $\mathbb{M}$ = Set of modules in a chain. A module is denoted as $m \in \mathbb{M}$

  $\mathbb{DAG}$ = A chain is represented as a DAG. A DAG is also called Precedence Graph, which can be represented as:

$$\mathbb{DAG}_{l,m} = \begin{cases} 1, & \text{if module } l \in \mathbb{M} \text{ directly precedes } m \in \mathbb{M} \\ 0, & \text{otherwise} \end{cases}$$

  $\mathbb{G}$ = Available GPU machines. **Note:** All instances of GPU machines are indexed uniquely.

  Example - If $\mathbb{G} = \{1080\text{Ti}_1,\ 1080\text{Ti}_2,\ ...,\ 2080\text{Ti}_1,\ 2080\text{Ti}_2,\ 2080\text{Ti}_3\}$ denotes 2×1080Ti machines and 3×1080Ti machines among other GPUs.

  P = List of offline profiles of each module on instances of GPU machines.

  $P_{m,g}$ = List of profiles of module $m \in \mathbb{M}$ on a GPU instance $g \in \mathbb{G}$.

  $P^k_{m,g} = \{< b^k_{m,g}, p^k_{m,g}, l^k_{m,g}, d^k_{m,g} >\}$, where $P^k_{m,g}$ denotes one batch-parallel-latency configuration.

  C_g = List of cost of running a GPU instance $g \in \mathbb{G}$.

  $S_{l,m}$ = Scaling factor from module $l \in \mathbb{M}$ to $m \in \mathbb{M}$. Namely, each input to module l will generate $S_{l,m}$ outputs, which will serve as inputs to module m.

  R_m = Input rate of module $m \in \mathbb{M}$

$$R_m = \sum_{\substack{l \in \mathbb{M} \\ \mathbb{DAG}_{l,m}=1}} R_l \times \mathrm{S}_{l,m}$$

$$R_{m0} = \Omega_{in}, \text{ where } \Omega_{in} \text{ is input rate to source module } m0$$

  L_{SLO} = Latency SLO

- *Decision Variables:*

  $X^k_{m,g}$: Allocation variable

$$X^k_{m,g} = \begin{cases} 1, & \text{if module } m \in \mathbb{M} \text{ is scheduled on } g \in \mathbb{G} \text{ with profile configuration } P^k_{m,g} \\ 0, & \text{otherwise} \end{cases}$$

  $R^k_{m,g}$: Input rate of a model m on GPU g with profile configuration $P^k_{m,g}$

$$R^k_{m,g} = \begin{cases} > 0, & \text{if } X^k_{m,g} = 1 \\ 0, & \text{otherwise} \end{cases}$$

  $U^k_{m,g}$: Capacity allocated to a model m on GPU g with profile configuration $P^k_{m,g}$

$$U^k_{m,g} = \begin{cases} \frac{R^k_{m,g}}{\frac{b^k_{m,g} \times p^k_{m,g}}{d^k_{m,g}}}, & \text{if } X^k_{m,g} = 1 \\ 0, & \text{otherwise} \end{cases}$$

  ST_m = Starting time for executing module m. This variable is required to calculate latency along critical path in the $\mathbb{DAG}$.

  L_{max} = Critical latency denoted by sum of latency of modules on the critical path (longest path) of the DAG. Thus, critical latency is the max of latency along all paths of the DAG from source to sink module.

- *Objective Function:*

  Minimize cost of running all modules of a chain on GPU instances $\mathbb{G}$.

$$\min \sum_{m \in \mathbb{M}} \sum_{g \in \mathbb{G}} C_g \times \sum_{k=0}^{|P_{m,g}|} X_{m,g}^k$$

- *Constraints:*

  1. A module $m \in \mathbb{M}$ running on gpu instance $g \in \mathbb{G}$ can take a unique offline profile configuration.

$$\sum_{k=0}^{|P_{m,g}|} X_{m,g}^k \leq 1 \qquad \forall (m \in \mathbb{M}, g \in \mathbb{G})$$

  2. Total input rate of module $m \in \mathbb{M}$ running on all the gpu instances should equal to R_m.

$$\sum_{g \in \mathbb{G}} \sum_{k=0}^{|P_{m,g}|} X_{m,g}^k \times R_{m,g}^k = R_m \qquad \forall (m \in \mathbb{M})$$

  3. Total capacity allocated to all the models running on a gpu instance g should be bounded by 1.0.

$$\sum_{m \in \mathbb{M}} \sum_{k=0}^{|P_{m,g}|} U_{m,g}^k \leq 1.0 \qquad \forall (g \in \mathbb{G})$$

  4. If module $l \in \mathbb{M}$ precedes $m \in \mathbb{M}$, the module m can process an input after module l has produced output for the same input.

$$ST_m \geq ST_l + X_{l,g}^k \times l_{l,g}^k \qquad \forall (l \in \mathbb{M}, m \in \mathbb{M}, 0 \leq k \leq |P_{l,g}|, g \in \mathbb{G}), \text{ where } \mathbb{DAG}_{l,m} = 1$$

  5. Critical latency L_{max} denoted by sum of latency of modules on the critical path (longest path) of the DAG.

$$L_{max} \geq ST_m + X_{m,g}^k \times l_{m,g}^k \qquad \forall (l \in \mathbb{M}, m \in \mathbb{M}, 0 \leq k \leq |P_{m,g}|, m \in \mathbb{M}, g \in \mathbb{G})$$

  6. Critical latency should be within latency SLO L_{SLO}

$$L_{max} \leq L_{\text{SLO}}$$