A MILP Formulation

A.1 General Formulation

Our general model describes total throughput which depends on the placement decision, core allocation, and branching. "Branching" represents a situation when an NF could have more than one downstream NF. The model can be decomposed into smaller pieces, which is concerned with throughput at different levels of the system: subgroup throughput, branch throughput, section throughput, chain throughput, and total throughput. Each section contains a complete set of branches that starts at the beginning and finishes at the end of the section. The notations are listed in Table 1.

$$\sum_{n\in\mathcal{N}} \underbrace{\min_{s\in\mathcal{S}_n} \underbrace{\sum_{b\in\mathcal{B}_{ns}} \underbrace{W_{nsb} \underbrace{\sum_{p\in\mathcal{P}_{nsb}} \underbrace{I^p_{nsb} \underbrace{\min_{k\in p} \underbrace{\left[\frac{\texttt{CPU\_FREQUENCY}}{\sum_{j\in k} \frac{C^j_{nsb}}{1+\mathbb{I}[j \text{ is stateless}](m^k_{nsb}-1)}}\right]}_{\text{subgroup throughput}}}_{\text{branch throughput}}}_{\text{section throughput}}}_{\text{chain throughput}}}_{\text{total throughput}}$$

Subgroup throughput is the inverse of aggregated CPU cycles for a run-to-completion subgroup. Multiple cores can be allocated to subgroup to increase parallel throughput. Stateless NFs benefit from multi-core allocation, but stateful NFs do not. For branch-level (including non-branch), each branch can have many subgroups and we find the minimal number across subgroups, conceptually representing the bottleneck of the branch. For section level, we identify several branches together in a section and what percentage of paths traverse a branch, yielding an average throughput for the whole section. At the chain level, there may be many sections and we pick the section that has the minimal throughput. Finally, we sum the throughput from different chains to find out the optimal placement based upon total throughput.

This throughput depends on the decision of NF placement and core allocation. This decision has to be restricted by the following constraints.

Only one placement option is selected at each branch:

$$\sum_{p\in\mathcal{P}_{nsb}} I^p_{nsb} = 1 \quad \forall\, n\in\mathcal{N}, s\in\mathcal{S}_n, b\in\mathcal{B}_{ns}$$

.

Only subgroups in a selected placement get CPU cores:

$$m^k_{nsb} \le M \sum_{p:\{p\ni k\}} I^p_{nsb} \quad \forall\, n\in\mathcal{N}, s\in\mathcal{S}_n, b\in\mathcal{B}_{ns}, k\in\mathcal{K}_{asb}$$

Only cores from the same CPU are allocated to a subgroup:

$$m^k_{nsb} = \sum_{u\in\mathcal{U}} v^{ku}_{nsb}, \quad \sum_{u\in\mathcal{U}} I^{ku}_{nsb} = 1, \quad v^{ku}_{nsb} \le I^{ku}_{nsb} M_u$$
$$\forall\, n\in\mathcal{N}, s\in\mathcal{S}_n, b\in\mathcal{B}_{ns}, k\in\mathcal{K}_{asb}, u\in\mathcal{U}$$

Notations representing the system (given)	
$\mathcal{N}$	set of NF chains
$\mathcal{S}_n$	set of sections in chain $n\in\mathcal{N}$
$\mathcal{B}_{ns}$	set of branches in section $s\in\mathcal{S}_n$
$\mathcal{K}_{asb}$	set of all possible subgroups within brach $b\in\mathcal{B}_{ns}$
$\mathcal{P}_{nsb}$	set of all possible placements of branch $b\in\mathcal{B}_{ns}$
$\mathcal{U}$	set of CPUs
C^j_{nsb}	number of cycles of function j in branch $b\in\mathcal{B}_{ns}$
M_u	number of cores in CPU $u\in\mathcal{U}$
Decision variables	
I^p_{nsb}	indication of selecting placement $p\in\mathcal{P}_{nsb}$
I^{ku}_{nsb}	indication of allocating core from CPU $u\in\mathcal{U}$ to subgroup $k\in\mathcal{K}_{asb}$
m^k_{nsb}	number of cores allocated to subgroup $k\in\mathcal{K}_{asb}$
v^{ku}_{nsb}	number of cores from CPU $u\in\mathcal{U}$ allocated to subgroup $k\in\mathcal{K}_{asb}$

Table 1: Optimization Model Notations and Variables

Limited CPU cores:

$$\sum_{n\in\mathcal{N}}\sum_{s\in\mathcal{S}_n}\sum_{b\in\mathcal{B}_{ns}}\sum_{k\in\mathcal{K}_{asb}} v^{ku}_{nsb} \le M_u \quad \forall\, u\in\mathcal{U}$$

Lastly, we have general variable constraints that $m^k_{nsb}, v^{ku}_{nsb} \in \mathbb{Z}_+$ and $I^p_{nsb}, I^{ku}_{nsb} \in \{0,1\}$.

This formulation is a mixed-integer (binary) linear optimization with the number of indicator variables that exponentially increases with the number of network functions. It is possible to extend this formulation to include subgroup sharing among NF chains, but we present this formulation for clarity.

A.2 Linear Program Formulation

The MILP formulation in §A.1 does not scale. We propose our heuristic algorithm for scalability. The heuristic algorithm carefully chooses a placement option consisting of NFs, NICs and CPU cores and evaluates it using a linear program that maximizes aggregate throughput subject to SLO, link capacity, and latency constraints.

Let $\mathcal{N}$ be an index set of chains. Chain n is decomposed to linear chains. Let $\mathcal{B}_n$ be the indices of these linear chains. Given a placement of NFs, we know all subgroups of a chain. We index subgroups of chain n by an index set $\mathcal{S}_n$. The weighted cycle of chain n's subgroup s is $Y_{ns} = \sum_{k\in s} Y^k \frac{\sum_{b\in\mathcal{B}_n, b\in k} W_{nb}}{\sum_{b\in\mathcal{B}_n, b\in s} W_{nb}}$, where Y^k is the cycle number of NF k and W_{nb} is the weight of a linear chain (n,b). We use notation $b\in s$ (or $b\in k$) to represent that a linear chain b is using subgroup s (or NF k). The weight of a linear chain is the product of all weights along the chain. If a subgroup (n,s) is replicated by M CPU cores, its achievable throughput is $T_{ns} = \frac{M}{Y_{ns}}$.

Further, a given placement determines the number of times a linear chain (n,b) traverses over a NIC i. This number is denoted by R^i_{nb}. The capacity of NIC i is C_i, and we use I as the index set of these NICs. A linear program that maximizes aggregate (marginal) throughput subject to SLO, link capacity,

and latency constraints is formulated below. The SLO is the requirements of minimum and maximum throughput denoted respectively by $T_n^{\min}$ and $T_n^{\max}$ for every chain n. Decision variable t_n denotes throughput of chain n.

$$
\begin{aligned}
\text{Maximize} \quad & \sum_{n \in \mathcal{N}} t_n \\
\text{Subject to} \quad & t_n W_{nb} \leq \min_{s \in b} \{T_{ns}\} && \forall n \in \mathcal{N}, b \in \mathcal{B}_n \\
& \sum_{n \in \mathcal{N}} \sum_{b \in \mathcal{B}_n} t_n W_{nb} R^i_{nb} \leq C_i && \forall i \in I \\
& \sum_{s \in b} Y_{ns} \leq L_n && \forall n \in \mathcal{N}, b \in \mathcal{B}_n \\
& t_n \in [T_n^{\min}, T_n^{\max}] && \forall n \in \mathcal{N}
\end{aligned}
$$

The first constraint represents the maximum achievable throughput of linear chain (n,b) that is limited by some subgroup's throughput. The second constraint limits chain's throughput and bouncing by NIC capacity. The last two constraints bound latency and SLO, where L_n is the maximum latency of chain n.

A.3 Heuristic Complexity

Suppose that there are K NFs in total to be deployed into our P heterogeneous NF execution platforms where $P \geq 2$. The core difference between brute-force and heuristic algorithms lies in the number of placement options in consideration. The brute-force placement algorithm considers $O(P^K)$ placement options while Lemur's heuristic placement algorithm consider $O(K^2)$ options. For each NF, brute-force algorithm needs to consider P different platforms to be deployed. On the contrary, the heuristic algorithm initiates with hardware-preferred placement with $O(1)$ complexity and gradually coalesces subgroups. The coalescing depends on the number of subgroup pairs, which grow as $O(K^2)$, so the overall complexity of the heuristic algorithm is $O(K^2)$.