A RAGGED BATCHING IN DECODE.

Ragged Batching in Decode. For the purpose of our evaluations, we quantify the heterogeneity of a ragged batch as the ratio of average context length to the maximum context length present in the batch. Figure 11 shows the speedup of LA over FD. We observe that as the heterogeneity of batch increases, LA delivers a higher speedup because of better distribution of work across SMs.

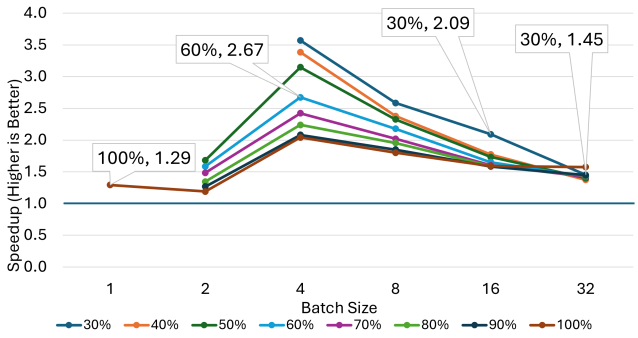

Figure 11. Speedup offered by LA over FD at different batch sizes with heterogeneous context lengths. Batch context ratio(%) shows the ratio of average context length over maximum context length

B LEANTILE ALGORITHM

Algorithm 1 LeanTile() for a sequence of lean tile iterations.

function LeanTile(tile_idx, iter_begin, iter_end)
shared $O_{acc}[T_m, d]$
shared $Q_f[T_m, d]$
shared $K_f[T_n, d]$
shared $V_f[T_n, d]$
shared $m[T_m, 1]$
shared $l[T_m, 1]$
Initialize O_{acc} to $(0)_{T_m \times d} \in R^{T_m \times d}$ in SMEM.
Initialize m to $(-\infty)_{T_m \times 1}$ and l to $(0)_{T_m \times 1} \in R^{T_m \times 1}$ in SMEM.
$mm = T_m \times$ (tile_idx / 1)
$nn = d \times$ (tile_idx % 1)
Perform lean tile iterations for this output tile.
for $iter = iter\_begin$ **to** $iter\_end$ **do**
    $kk = iter \times T_n$
    load fragments from GMEM to SMEM
    $Q_f = LoadFragment(Q, mm, nn)$
    $K_f = LoadFragment(K, nn, kk)$
    $V_f = LoadFragment(V, nn, kk)$
    Compute on chip:
    $S_f = Q_f K_f$ where $S_f \in R^{T_m \times T_n}$
    $m^{new} = max(m, rowmax(S_f))$
    $P_f = exp(S_f - m^{new})$ where $P_f \in R^{T_m \times T_n}$
    $l^{new} = e^{m - m^{new}} l + rowsum(P_f)$
    $O_{acc} = P_f V_f + diag(e^{m - m^{new}}) O_{acc}$
    $l = l^{new}, m = m^{new}$
end for
return O_{acc}, l, m
end function

C PARTITION STRATEGY

Without loss of generality, we describe this process of reduction to obtain one row vector of the attention score matrix $\mathbf{S}$, of the form $\begin{bmatrix}\mathbf{S}^{(x)} & \mathbf{S}^{(y)}\end{bmatrix}$ consisting of some unequal length vectors $\mathbf{S}^{(x)}, \mathbf{S}^{(y)}$ where $\mathbf{S}^{(x)} \in \mathbb{R}^{1\times B_c^{(x)}}$ and $\mathbf{S}^{(y)} \in \mathbb{R}^{1\times B_c^{(y)}}$, where 1 is the query length and $B_c^{(x)}$ and $B_c^{(y)}$ are the unequal context lengths. The vectors $\mathbf{S}^{(x)}$ and $\mathbf{S}^{(y)}$ were computed from $\mathbf{Q}\times(\mathbf{K}^{(x)})^T$ and $\mathbf{Q}\times(\mathbf{K}^{(y)})^T$ (illustrated Figure in Appendix C). Note that, to generalize this procedure for blocks of any size, the context length of $\mathbf{K}^{(x)}$ and $\mathbf{K}^{(y)}$ are $B_c^{(x)}$ and $B_c^{(y)}$ and are not necessarily equal.

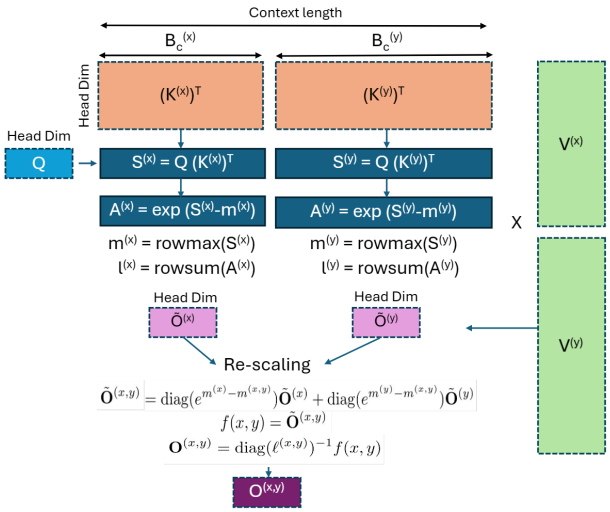

Figure 12. Illustrative diagram showing LeanAttention's partitioning strategy with two differently sized work volumes of a head assigned to different CTAs. The un-scaled outputs are independently computed and re-scaled later in a reduction operation. Note that this can be generalized to any arbitrary-sized work volume split.

D PROOF OF ASSOCIATIVITY

Proving that $f(f(x,y),z) = f(x,y,z)$:

$$
\begin{aligned}
f(x,y) &= \tilde{\mathbf{O}}^{(x,y)} \\
f(f(x,y),z) &= \mathrm{diag}(e^{m^{(x,y)} - m^{((x,y),z)}})\tilde{\mathbf{O}}^{(x,y)} \\
&\quad + \mathrm{diag}(e^{m^{(z)} - m^{((x,y),z)}})\tilde{\mathbf{O}}^{(z)} \\
&= \mathrm{diag}(e^{m^{(x,y)} - m^{(x,y,z)}})\tilde{\mathbf{O}}^{(x,y)} \\
&\quad + \mathrm{diag}(e^{m^{(z)} - m^{(x,y,z)}})\tilde{\mathbf{O}}^{(z)} \\
&= \mathrm{diag}(e^{m^{(x,y)} - m^{(x,y,z)}}) \\
&\quad \times (\mathrm{diag}(e^{m^{(x)} - m^{(x,y)}})\tilde{\mathbf{O}}^{(x)} \\
&\quad + \mathrm{diag}(e^{m^{(y)} - m^{(x,y)}})\tilde{\mathbf{O}}^{(y)}) \\
&\quad + \mathrm{diag}(e^{m^{(z)} - m^{(x,y,z)}})\tilde{\mathbf{O}}^{(z)} \\
&= \mathrm{diag}(e^{m^{(x)} - m^{(x,y,z)}})\tilde{\mathbf{O}}^{(x)} \\
&\quad + \mathrm{diag}(e^{m^{(y)} - m^{(x,y,z)}})\tilde{\mathbf{O}}^{(y)} \\
&\quad + \mathrm{diag}(e^{m^{(z)} - m^{(x,y,z)}})\tilde{\mathbf{O}}^{(z)} \\
&= \tilde{\mathbf{O}}^{(x,y,z)} = f(x,y,z)
\end{aligned}
$$

Therefore, $f(f(x,y),z) = f(x,y,z)$ and similarly $\ell^{((x,y),z)} = \ell^{(x,y,z)}$. For brevity, we omit the proof of $f(x, f(y,z)) = f(x,y,z)$, but it can deduced in a similar manner.

This associativity of softmax re-scaling is leveraged in LeanAttention to concurrently calculate the "partial" outputs produced from unequally sized KV blocks and then "reduce" them to obtain exact attention.

E DETAILED ALGORITHM OF LEANATTENTION

Algorithm 2 Lean Attention

shared $O[T_m, d]$
shared $m[T_m, 1]$
shared $l[T_m, 1]$
Number of output tiles: $C_m = \lceil N_q / T_m \rceil$
Number of iterations for each output tile: $C_n = \lceil N_k / T_n \rceil$
Total number of iterations: $I = C_m C_n$
Number of iterations per CTA: $I_G = I/G$
fork CTA_g in G **do**
cta_start = g I_G and cta_end = cta_start + I_G
for iter = cta_start **to** cta_end **do**
    Index of current output tile: tile_idx = iter / C_n
    tile_iter = tile_idx $\times C_n$
    tile_iter_end = tile_iter + C_n
    local_iter = iter - tile_iter
    local_iter_end = min(tile_iter_end, cta_end) - tile_iter
    O, m, l = LeanTile(tile_idx, local_iter, local_iter_end)
    host-block if: iter == tile_iter
    finishing-block if: cta_end >= tile_iter_end
    if !(host-block) **then**
        StorePartials(Op[g], O)
        StorePartials(mp[g], m)
        StorePartials(lp[g], l)
        Signal(flags[g])
    else
        if !(finishing-block) **then**
            last_cta = tile_iter_end / C_n
            for cta = (g + 1) **to** last_cta **do**
                Wait(flags[cta])
                m_{cta} = LoadPartials(mp[cta])
                l_{cta} = LoadPartials(lp[cta])
                O_{cta} = LoadPartials(Op[cta])
                $m^{new} = max(m_{cta}, m)$
                $l^{new} = e^{m_{cta} - m^{new}} l_{cta} + e^{m - m^{new}} l$
                $O^{new} = e^{m_{cta} - m_i^{new}} O_{cta} + e^{m - m_i^{new}} O$
                Update $m = m_i^{new}, l = l_i^{new}$
            end for
        end if
        Write $O = diag(l)^{-1} O$ to GMEM.
        Write $L = m + log(l)$ to GMEM.
    end if
    iter = tile_iter_end
end for
join

F Energy Consumption Evaluation

Fixed-split partitioning results in imbalanced workloads across the SMs, leaving many of them idle during the final stages of computation. This inefficiency makes fixed-split attention mechanisms energy-inefficient. As shown in Figure 13, the disparity in energy consumption between FlashDecoding, FlashInfer, and LeanAttention increases as context lengths grow over 128k. LeanAttention, with its well-balanced load partitioning strategy, ensures more consistent utilization of SMs, making it significantly more energy-efficient.

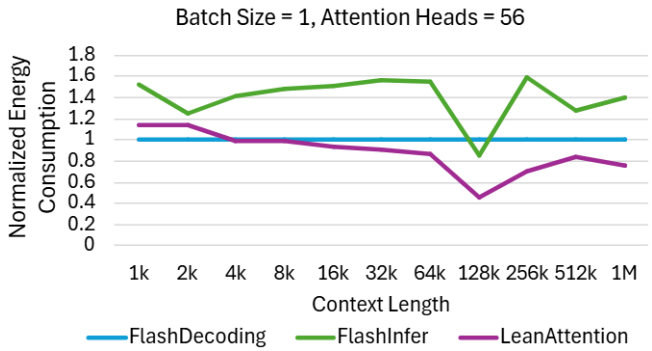

Figure 13. Ratio of Energy consumed by attention kernel to energy consumed by FlashDecoding kernel of different attention mechanisms for batch size = 1, number of heads = 56, head dimension = 64 on a single Nvidia-A100-80GB GPU as measured using NVML APIs.