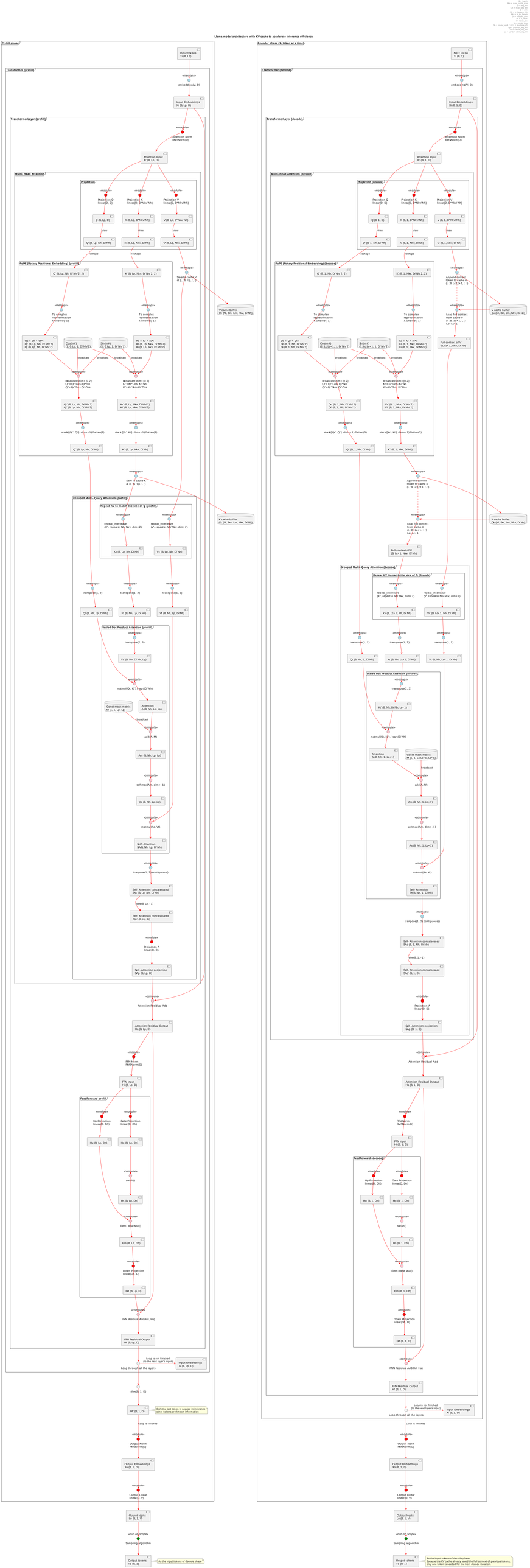
Llama model architecture with KV cache to accelerate inference efficiency
Prefill phase
Input tokens
Ti (B, Lp)
«tensorop»
embedding(V, D)
Input Embeddings
Xi (B, Lp, D)
Transformer (prefill)
TransformerLayer (prefill)
«tensorop»
Attention Norm
RMSNorm(D)
Attention Input
Xr (B, Lp, D)
Multi-Head Attention
Projection
«tensorop»
Projection Q
linear(D, D)
«tensorop»
Projection K
linear(D, D*Nkv/Nh)
«tensorop»
Projection V
linear(D, D*Nkv/Nh)
Q (B, Lp, D)
K (B, Lp, D*Nkv/Nh)
V (B, Lp, D*Nkv/Nh)
«view»
Q' (B, Lp, Nh, D/Nh)
«view»
K' (B, Lp, Nkv, D/Nh)
«view»
V' (B, Lp, Nkv, D/Nh)
«reshape»
RoPE (Rotary Positional Embedding) (prefill)
Q' (B, Lp, Nh, D/Nh/2, 2)
K' (B, Lp, Nkv, D/Nh/2, 2)
«tensorop»
To complex representation x.unbind(-1)
Qc = Qr + Qi*i
Q' (B, Lp, Nh, D/Nh/2)
Q' (B, Lp, Nh, D/Nh/2, 2)
Cos(m*f)
[1, 0'(p), 1, D/Nh/2]
Sin(m*f)
[0, 0'(p), 1, D/Nh/2]
Kc = Kr + Ki*i
K' (B, Lp, Nkv, D/Nh/2)
K' (B, Lp, Nkv, D/Nh/2, 2)
«broadcast»
«broadcast»
«broadcast»
«broadcast»
«tensorop»
Broadcast dim=[0,2]
Qr*Cos-Qi*Sin
Qi*Cos+Qr*Sin
«tensorop»
Broadcast dim=[0,2]
Kr*Kr*Cos-Ki*Sin
Ki*Kr*Sin+Ki*Cos
Q0' (B, Lp, Nh, D/Nh/2)
Q1' (B, Lp, Nh, D/Nh/2)
K0' (B, Lp, Nkv, D/Nh/2)
K1' (B, Lp, Nkv, D/Nh/2)
«tensorop»
stack[Q0', Q1', dim=-1].flatten(3)
«tensorop»
stack[K0', K1', dim=-1].flatten(3)
Q' (B, Lp, Nh, D/Nh)
K' (B, Lp, Nkv, D/Nh)
«tensorop»
Save to cache V
at [:, :B, :Lp, :, :]
V cache buffer
Cv (M, Bm, Lm, Nkv, D/Nh)
«tensorop»
Save to cache K
at [:, :B, :Lp, :, :]
K cache buffer
Ck (M, Bm, Lm, Nkv, D/Nh)
Grouped Multi-Query Attention (prefill)
Repeat KV to match the size of Q (prefill)
«tensorop»
repeat_interleave
(K', repeats=Nh/Nkv, dim=2)
«tensorop»
repeat_interleave
(V', repeats=Nh/Nkv, dim=2)
Kv (B, Lp, Nh, D/Nh)
Vv (B, Lp, Nh, D/Nh)
«tensorop»
transpose(1, 2)
«tensorop»
transpose(1, 2)
«tensorop»
transpose(1, 2)
Qt (B, Nh, Lp, D/Nh)
Kt (B, Nh, Lp, D/Nh)
Vt (B, Nh, Lp, D/Nh)
Scaled Dot Product Attention (prefill)
«tensorop»
transpose(2, 3)
Kt' (B, Nh, D/Nh, Lp)
«compute»
matmul(Qt, Kt') / sqrt(D/Nh)
Const mask matrix
M (1, 1, Lp, Lp)
Attention
A (B, Nh, Lp, Lp)
«broadcast»
«compute»
add(A, M)
Am (B, Nh, Lp, Lp)
«compute»
softmax(Am, dim=-1)
As (B, Nh, Lp, Lp)
«compute»
matmul(As, Vt)
Sa0: Attention
SA(B, Nh, Lp, D/Nh)
«tensorop»
transpose(1, 2).contiguous()
Sa1: Attention concatenated
SAc (B, Lp, Nh, D/Nh)
«view(B, Lp, - 1)»
Sa2: Attention concatenated
SAc' (B, Lp, D)
«compute»
Projection A
linear(D, D)
Sa3: Attention projection
SAp (B, Lp, D)
«compute»
Attention Residual Add
Attention Residual Output
Ha (B, Lp, D)
«tensorop»
FFN Norm
RMSNorm(D)
FFN Input
Hi (B, Lp, D)
FeedForward prefill
«compute»
Up Projection
linear(D, DH)
«compute»
Gate Projection
linear(D, DH)
Hu (B, Lp, DH)
Hg (B, Lp, DH)
«compute»
swish()
Hs (B, Lp, DH)
«compute»
Elem-Wise Mul()
Hm (B, Lp, DH)
«compute»
Down Projection
linear(DH, D)
Hd (B, Lp, D)
«compute»
FNN Residual Add(Hd, Ha)
FFN Residual Output
Hf (B, Lp, D)
Loop is not finished
(to the next layer's input)
Input Embeddings
Xi (B, Lp, D)
Loop through all the layers
«loop(L - 1, D)»
Hf' (B, L, D)
Only the last token is needed in inference
other tokens are known information
Loop is finished
«tensorop»
Output Norm
RMSNorm(D)
Output Embeddings
Xo (B, L, D)
«tensorop»
Output Linear
linear(D, V)
Output logits
Lo (B, L, V)
«out-of-scope»
Sampling algorithm
Output tokens
To (B, L)
As the input tokens of decode phase
Decode phase (L - token at a time)
Next token
Tt (B, 1)
«tensorop»
embedding(V, D)
Input Embeddings
Xi (B, 1, D)
Transformer (decode)
TransformerLayer (decode)
«tensorop»
Attention Norm
RMSNorm(D)
Attention Input
Xr (B, 1, D)
Multi-Head Attention (decode)
Projection (decode)
«tensorop»
Projection Q
linear(D, D)
«tensorop»
Projection K
linear(D, D*Nkv/Nh)
«tensorop»
Projection V
linear(D, D*Nkv/Nh)
Q (B, 1, D)
K (B, 1, D*Nkv/Nh)
V (B, 1, D*Nkv/Nh)
«view»
Q' (B, 1, Nh, D/Nh)
«view»
K' (B, 1, Nkv, D/Nh)
«view»
V' (B, 1, Nkv, D/Nh)
«reshape»
RoPE (Rotary Positional Embedding) (decode)
Q' (B, 1, Nh, D/Nh/2, 2)
K' (B, 1, Nkv, D/Nh/2, 2)
«tensorop»
To complex representation x.unbind(-1)
Qc = Qr + Qi*i
Q' (B, 1, Nh, D/Nh/2)
Q' (B, 1, Nh, D/Nh/2, 2)
Cos(m*f)
[1, Lc:Lc+1, 1, D/Nh/2]
Sin(m*f)
[1, Lc:Lc+1, 1, D/Nh/2]
Kc = Kr + Ki*i
K' (B, 1, Nkv, D/Nh/2)
K' (B, 1, Nkv, D/Nh/2, 2)
«broadcast»
«broadcast»
«broadcast»
«broadcast»
«tensorop»
Broadcast dim=[0,2]
Qr*Cos-Qi*Sin
Qi*Cos+Qr*Sin
«tensorop»
Broadcast dim=[0,2]
Kr*Kr*Cos-Ki*Sin
Ki*Kr*Sin+Ki*Cos
Q0' (B, 1, Nh, D/Nh/2)
Q1' (B, 1, Nh, D/Nh/2)
K0' (B, 1, Nkv, D/Nh/2)
K1' (B, 1, Nkv, D/Nh/2)
«tensorop»
stack[Q0', Q1', dim=-1].flatten(3)
«tensorop»
stack[K0', K1', dim=-1].flatten(3)
Q' (B, 1, Nh, D/Nh)
K' (B, 1, Nkv, D/Nh)
«tensorop»
Append current token to cache V
[:, B, Lc:Lc+1, :, :]
«tensorop»
Load full context from cache V
[:, B, :Lc+1, :, :]
Lc+1=Lc+1
V cache buffer
Cv (M, Bm, Lm, Nkv, D/Nh)
Full context of V
(B, Lc+1, Nkv, D/Nh)
«tensorop»
Append current token to cache K
[:, B, Lc:Lc+1, :, :]
«tensorop»
Load full context from cache K
[:, B, :Lc+1, :, :]
Lc+1=Lc+1
K cache buffer
Ck (M, Bm, Lm, Nkv, D/Nh)
Full context of K
(B, Lc+1, Nkv, D/Nh)
Grouped Multi-Query Attention (decode)
Repeat KV to match the size of Q (decode)
«tensorop»
repeat_interleave
(K', repeats=Nh/Nkv, dim=2)
«tensorop»
repeat_interleave
(V', repeats=Nh/Nkv, dim=2)
Kv (B, Lc+1, Nh, D/Nh)
Vv (B, Lc+1, Nh, D/Nh)
«tensorop»
transpose(1, 2)
«tensorop»
transpose(1, 2)
«tensorop»
transpose(1, 2)
Qt (B, Nh, 1, D/Nh)
Kt (B, Nh, Lc+1, D/Nh)
Vt (B, Nh, Lc+1, D/Nh)
Scaled Dot Product Attention (decode)
«tensorop»
transpose(2, 3)
Kt' (B, Nh, D/Nh, Lc+1)
«compute»
matmul(Qt, Kt') / sqrt(D/Nh)
Attention
A (B, Nh, 1, Lc+1)
Const mask matrix
M (1, 1, Lc:Lc+1, Lc:Lc+1)
«broadcast»
«compute»
add(A, M)
Am (B, Nh, 1, Lc+1)
«compute»
softmax(Am, dim=-1)
As (B, Nh, 1, Lc+1)
«compute»
matmul(As, Vt)
Sa0: Attention
SA(B, Nh, 1, D/Nh)
«tensorop»
transpose(1, 2).contiguous()
Sa1: Attention concatenated
SAc (B, 1, Nh, D/Nh)
«view(B, 1, - 1)»
Sa2: Attention concatenated
SAc' (B, 1, D)
«compute»
Projection A
linear(D, D)
Sa3: Attention projection
SAp (B, 1, D)
«compute»
Attention Residual Add
Attention Residual Output
Ha (B, 1, D)
«tensorop»
FFN Norm
RMSNorm(D)
FFN Input
Hi (B, 1, D)
FeedForward (decode)
«compute»
Up Projection
linear(D, DH)
«compute»
Gate Projection
linear(D, DH)
Hu (B, 1, DH)
Hg (B, 1, DH)
«compute»
swish()
Hs (B, 1, DH)
«compute»
Elem-Wise Mul()
Hm (B, 1, DH)
«compute»
Down Projection
linear(DH, D)
Hd (B, 1, D)
«compute»
FNN Residual Add(Hd, Ha)
FFN Residual Output
Hf (B, 1, D)
Loop is not finished
(to the next layer's input)
Input Embeddings
Xi (B, 1, D)
Loop through all the layers
«loop(L - 1, D)»
Loop is finished
«tensorop»
Output Norm
RMSNorm(D)
Output Embeddings
Xo (B, 1, D)
«tensorop»
Output Linear
linear(D, V)
Output logits
Lo (B, 1, V)
«out-of-scope»
Sampling algorithm
Output tokens
Tt (B, 1)
As the input tokens of decode phase.
Because the KV cache already saved the full context of previous tokens,
only one token is needed for the next decode iteration.