Efficient Attention Mechanisms: BigBird and FlashAttention

Mohamed Ramadan

1 Introduction

Transformer models rely on the self-attention mechanism, where each token attends to every other token in the sequence.

Given a sequence of length n, full self-attention computes:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V$$

This requires computing an $n \times n$ attention matrix.

Problem:

- Time Complexity: $\mathcal{O}(n^2)$
- Memory Complexity: $\mathcal{O}(n^2)$

This becomes infeasible for long sequences (e.g., documents, DNA, video frames).

2 Full Attention Bottleneck

In full attention, every token interacts with every other token.

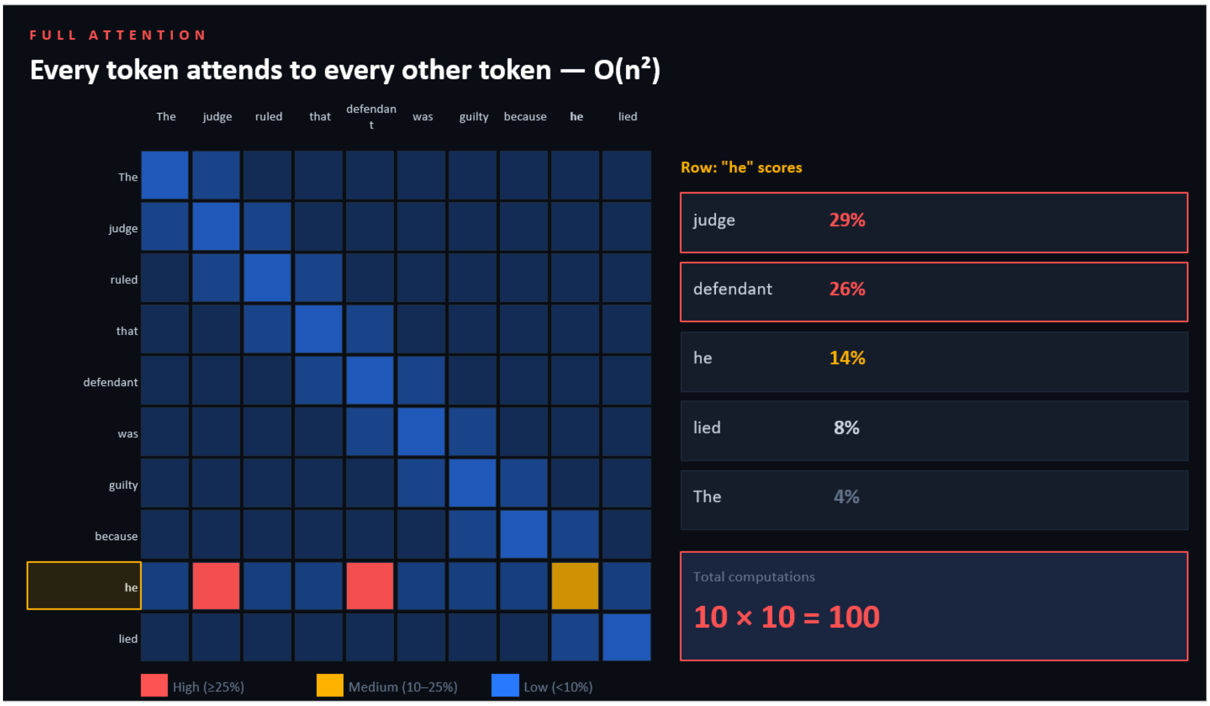

Figure 1: Full Self-Attention: Each token attends to all tokens ($\mathcal{O}(n^2)$).

For example, with 10 tokens:

$$10 \times 10 = 100 \text{ attention scores}$$

For 10,000 tokens:

$$10^4 \times 10^4 = 10^8$$

Clearly quadratic growth is the bottleneck.

3 BigBird: Sparse Attention for Linear Scaling

BigBird introduces structured sparse attention to reduce complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.
Instead of attending to all tokens, each token attends to:

- **Global tokens**
- **Local window tokens**
- **Random tokens**

$$\text{Complexity} = \mathcal{O}(n)$$

3.1 Intuition

- Global tokens maintain long-range communication.
- Local window captures nearby context.
- Random connections ensure theoretical expressivity.

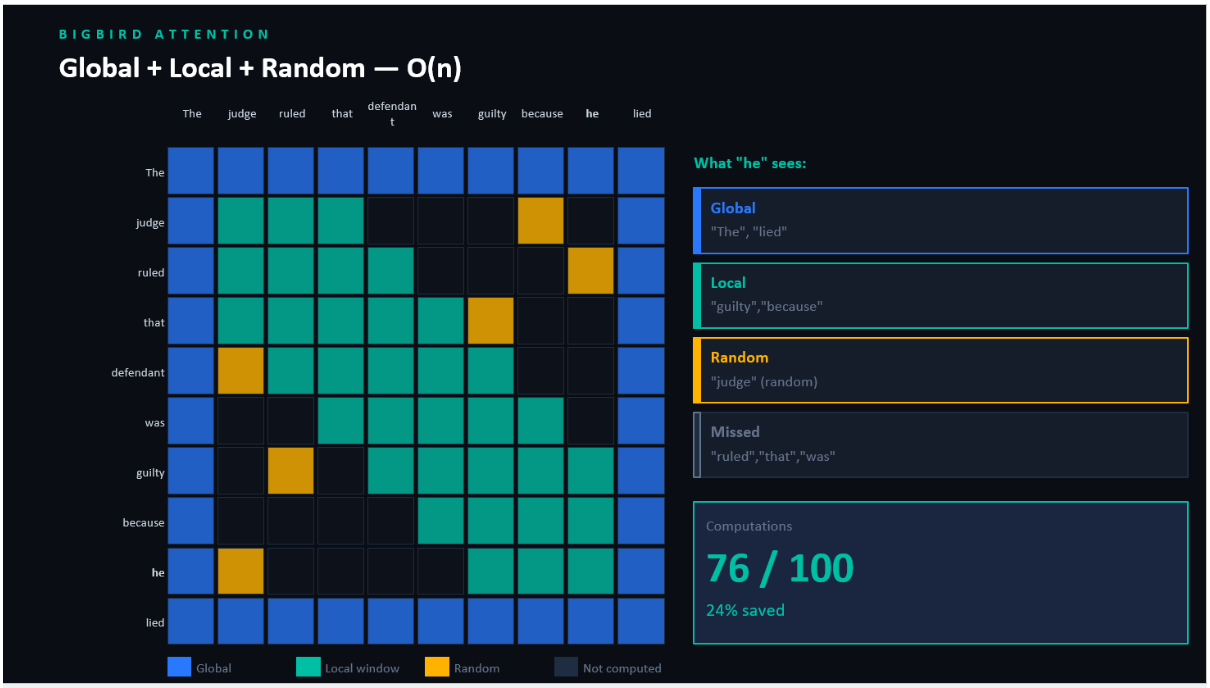

Figure 2: BigBird Sparse Attention Pattern (Global + Local + Random).

3.2 Why BigBird Works

BigBird was proven to be:

- A universal approximator of sequence functions
- Turing complete under certain conditions

Thus, it preserves theoretical power while reducing computational cost.

4 FlashAttention: IO-Aware Exact Attention

FlashAttention does **NOT** approximate attention.
Instead, it computes exact attention but optimizes memory access.

4.1 Core Idea

The real bottleneck is memory bandwidth (HBM), not FLOPs.
FlashAttention:

1. Tiles the attention matrix into blocks
2. Computes each block inside fast SRAM
3. Uses an online softmax trick to merge results
4. Avoids materializing the full $n \times n$ matrix

$$\text{Time Complexity: } \mathcal{O}(n^2)$$

$$\text{Memory Usage: Dramatically Reduced}$$

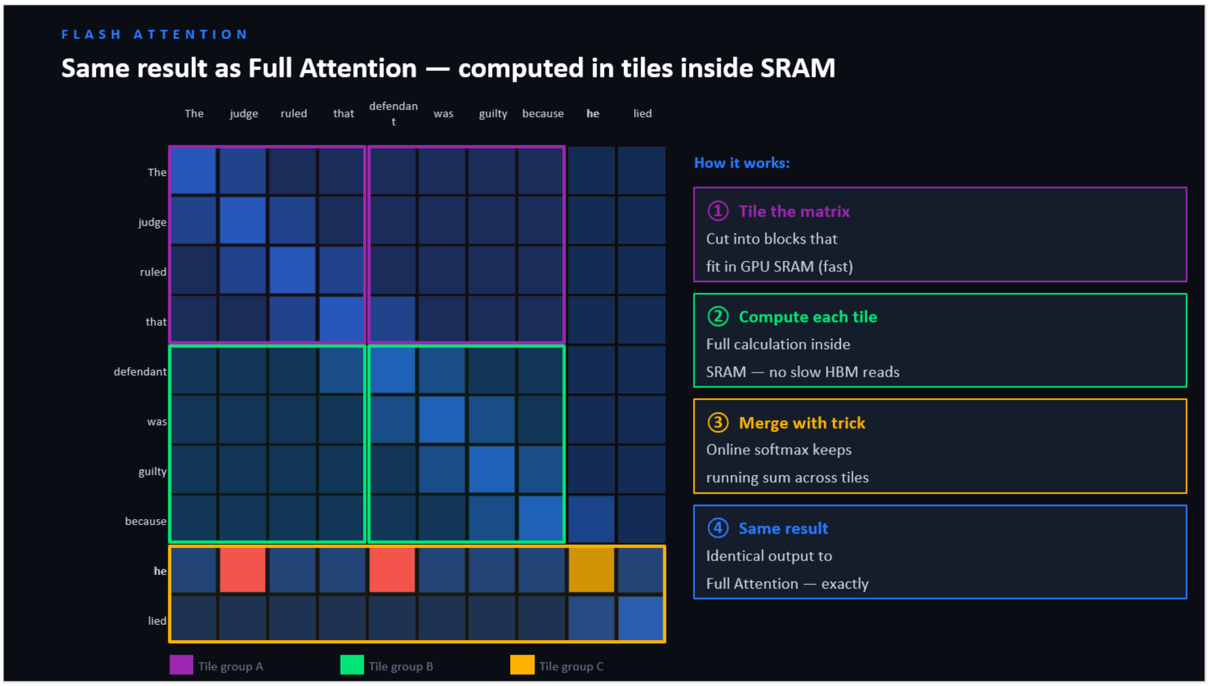

Figure 3: FlashAttention: Tiled computation inside GPU SRAM.

4.2 Key Insight: Online Softmax

Instead of computing:

$$\text{Softmax}(QK^T)$$

FlashAttention maintains a running max and running sum across tiles, ensuring numerical stability and exact equivalence to full attention.

Thus:

Same result as full attention, faster and memory efficient

5 BigBird vs FlashAttention

	BigBird	FlashAttention
Attention Type	Sparse	Exact
Time Complexity	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$
Memory Efficiency	High	Very High
Long Sequence Scaling	Excellent	Limited by quadratic FLOPs
Accuracy vs Full	Approximate	Identical
Best Use Case	Very long documents	Large LLM training/inference

Table 1: Comparison between BigBird and FlashAttention

6 When to Use Each?

Use BigBird When:

- Working with extremely long sequences (8k–100k tokens)
- You need linear scaling
- Small approximation is acceptable

Use FlashAttention When:

- Training large LLMs
- You want exact attention
- GPU memory bandwidth is bottleneck

7 Real-World Applications

BigBird Used In:

- Long document classification
- Question answering over long contexts
- Genomics

FlashAttention Used In:

- GPT-style models

- LLaMA variants
- Modern production LLM systems

8 Conclusion

Full attention is powerful but quadratic.

BigBird solves the scaling problem via sparse structure.

FlashAttention solves the memory bottleneck via IO-aware optimization.

They address different dimensions of the same core challenge.

$$\textbf{BigBird = Algorithmic Efficiency}$$

$$\textbf{FlashAttention = Hardware Efficiency}$$