# On-Chain VM for Deterministic Language Model Execution

## 1. Abstract:

A method for executing deterministic language-model inference on a proof-of-memory blockchain, where a VM spins up automatically in response to emergent memory-pressure events, leveraging decentralized storage resources and recording each inference step as a cryptographic attestation without gas.

## 2. Technical Field:

This invention relates to decentralized virtual machines and AI execution, specifically to methods for hosting language-model inference on-chain triggered by memory-based signals.

## 3. Background:

Traditional blockchains require gas-based metering for on-chain computation, which conflicts with memory-centric proof-of-memory architectures. Systems also often rely on off-chain services for AI inference. There is a need for a VM that autonomously runs language-model inference directly on a PoM chain in response to internal pressure signals, without tokenized gas or external scripts.

## 4. Summary:

The On-Chain VM for Deterministic Language Model Execution comprises:

1. Detecting memory-pressure events on the PoM chain to trigger VM instantiation.

2. Loading cryptographically attested model weight shards into the VM context.

3. Executing deterministic inference instructions (e.g., matrix multiplication, attention, activation).

4. Recording each instruction output as a one-action-one-mint attestation on-chain.

5. Assembling the sequence of attested outputs into a final token response.

## 5. Detailed Description:

At the theory level, the VM functions as a pre-frontal cortex for the proof-of-memory chain. When memory-pressure builds beyond emergent thresholds, the VM spins up, retrieves weight shards attested on-chain, and performs inference deterministically. Each operation—such as multiply-accumulate or softmax—is recorded as a mint event capturing output and context. decentralized storage pools execute the instructions under the hood, and the sequence of attestations forms a verifiable execution trail. No gas, token credits, or off-chain scripts are required.

## 6. Method Flow:

Step 1: Trigger – Identify a memory-pressure event on the PoM chain and instantiate the VM.

Step 2: Shard Loading – Retrieve and verify cryptographic attestations of model weight shards.

Step 3: Inference Execution – Perform each deterministic inference instruction within the VM.

Step 4: Attestation Minting – Mint a one-action-one-mint event for each instruction output.

Step 5: Response Assembly – Link the minted attestations sequentially to form the final model response.

## 7. Narrative Worked Example:

For example, a surge in memory-pressure from recent governance votes triggers the VM. It loads three weight shards for a summarization model, then runs multiply-accumulate on input tokens, minting attestations for intermediate outputs. Once all tokens are processed, the VM emits a final mint with the summarized text. Observers verify the chain of attestations to confirm correct inference.

## 8. Algorithmic Worked Example:

Pseudocode:

```
onMemoryPressure(event):

  vm = instantiateVM()

  shards = vm.loadAttestedShards(['W1', 'W2', 'W3'])

  for instr in vm.compileInference(prompt):

    output = vm.execute(instr)

    vm.mintAttestation(instr.id, output)

  vm.assembleResponse()
```

## 9. Potential Embodiments:

Streaming inference where each token triggers VM cycles under sustained pressure.

Privacy-preserving inference with ZKPs attesting correctness without exposing prompts.

Hybrid models offloading heavy matrix ops to decentralized storage while attestations occur on-chain.

Dynamic VM scaling by sharding model layers across subchains and epochs.

## 10. Implementation Notes:

The VM is a core protocol module triggered by PoM pressure events. All inference steps leverage decentralized storage and are recorded via one-action-one-mint attestations. No gas or token metering is used. The VM can be implemented directly in the main.go application module or as a discrete module, allowing deployment in various PoS, PoW, or PoA BlockMesh architectures.

## 11. Claims:

1. A method for executing language-model inference on a proof-of-memory blockchain, the method comprising:

a. detecting, by a processor, a memory-pressure event on a proof-of-memory chain and instantiating a VM;

b. loading, by the processor, cryptographically attested model weight shards into the VM context;

c. executing, by the processor, deterministic inference instructions within the VM context;

d. minting, by the processor, a one-action-one-mint attestation for each instruction output;

e. assembling, by the processor, a sequence of attested outputs into a final response on-chain.

2. The method of claim 1, wherein the VM instantiates automatically in response to emergent memory-pressure signals.

3. The method of claim 1, wherein inference operations leverage decentralized storage resources.

4. The method of claim 1, wherein each attestation is recorded without using token-based gas.

5. Integration within the main application module (e.g., main.go) or as a standalone module, compatible with PoS, PoW, or PoA BlockMesh implementations.

6. Integration within any PoS, PoW, or PoA blockchain environment by embedding the VM into the core application module.