Small Local AI, Large Memory: How a Memory Vault Can Make Efficient Models Far More Capable

Running artificial intelligence locally presents an important engineering trade-off.

Larger AI models usually provide stronger reasoning, better language understanding, more reliable coding assistance, and greater flexibility across unfamiliar subjects. However, they also require substantially more memory, processing power, storage, cooling, and electrical power.

Smaller models are easier to run privately on ordinary computers, but they cannot internally retain as much learned capability as their larger counterparts.

A well-designed **Memory Vault** offers a practical middle path.

Instead of trying to force an enormous model onto limited hardware, a local assistant can use a smaller model together with persistent memory, document retrieval, structured records, tools, verification systems, and carefully designed operating policies.

This does not magically transform a small model into a larger one. It does, however, allow the smaller model to become highly specialised, consistent, and useful within the user's own environment.

For systems such as **AEGIS**, the design principle is straightforward:

> Use the model for language, interpretation, and reasoning. Use the Memory Vault for persistent knowledge, project history, personal context, procedures, and verified records.

This article explains that principle, how the architecture works, where it genuinely closes the gap between model sizes, where it does not, and what hardware is realistically required to run local models from 1 billion to 70 billion parameters.

---

1. What Model Size Actually Means

AI model sizes are commonly described using a number followed by the letter **B**, meaning billions of parameters.

Examples include:

- 1B: approximately one billion parameters
- 3B: approximately three billion parameters
- 8B: approximately eight billion parameters
- 14B: approximately fourteen billion parameters
- 32B: approximately thirty-two billion parameters

- 70B: approximately seventy billion parameters

Parameters are numerical values learned during model training. They influence how the model processes language, recognises patterns, represents concepts, and predicts an appropriate response.

A larger parameter count often allows a model to represent more complex relationships. It may also improve reasoning, instruction following, coding, language quality, and performance on unfamiliar tasks.

However, parameter count is not a perfect measure of intelligence.

A newer 8B model trained on better data may outperform an older 13B or even 30B model in some areas. Techniques such as distillation can also transfer some capabilities from larger teacher models into smaller models. Meta, for example, has described using larger models as teachers when developing smaller 1B and 3B Llama models.

Model architecture, training data, post-training, context handling, tool support, quantisation quality, and prompting all matter.

Model size should therefore be treated as an important hardware and capability indicator, but not as a complete intelligence rating.

---

2. The Central Problem With Small Models

A smaller model has several practical advantages:

- It needs less RAM and VRAM.
- It starts more quickly.
- It generates responses faster on modest hardware.
- It consumes less power.
- It can run on laptops and small home servers.
- It is easier to keep completely local.
- It leaves resources available for other applications.

Its main disadvantage is that its internal capability ceiling is lower.

Compared with a strong larger model, a smaller model may be more likely to:

- misunderstand complicated instructions;
- lose track of multi-stage plans;
- make incorrect assumptions;
- struggle with unfamiliar programming problems;
- overlook subtle contradictions;
- mishandle large amounts of context;
- provide plausible but unsupported answers;
- fail to distinguish current records from outdated ones;
- produce inconsistent tool calls;

- require clearer prompts and stronger safeguards.

Adding memory does not remove these weaknesses. A Memory Vault does not alter the model's trained parameters or fundamentally increase its reasoning capacity.

What memory can do is reduce the amount of work the model must perform from memory alone.

---

3. What a Memory Vault Is

A Memory Vault is a persistent information system that exists outside the language model.

It may contain:

- user-approved memories;
- project records;
- software documentation;
- technical manuals;
- device inventories;
- source code summaries;
- household procedures;
- previous decisions;
- package versions;
- file paths;
- security policies;
- permissions;
- release histories;
- checksums;
- contacts;
- schedules;
- repair records;
- bookkeeping information;
- verified facts;
- archived conversations;
- current project status.

The model itself does not permanently remember this information merely because it has seen it in a previous conversation.

Instead, the information is stored in ordinary files, databases, indexes, or other controlled repositories. When the user asks a question, the assistant retrieves the most relevant records and temporarily places them into the model's working context.

This general technique is often called **retrieval-augmented generation**, or RAG.

The important distinction is:

> The model generates the response, but the Vault supplies the relevant evidence.

The Memory Vault is therefore not simply a large conversation transcript. A strong Vault is a managed knowledge system with structure, provenance, permissions, lifecycle controls, and retrieval logic.

---

4. The AEGIS Memory Principle

The AEGIS approach can be understood as a separation of responsibilities.

The model provides:

- natural-language understanding;
- interpretation of the user's request;
- limited reasoning and planning;
- summarisation;
- drafting;
- explanation;
- selection between available actions;
- conversion of retrieved facts into useful responses.

The Memory Vault provides:

- persistent storage;
- project continuity;
- canonical facts;
- previous decisions;
- approved preferences;
- procedures;
- historical records;
- document knowledge;
- source references;
- integrity information;
- status and version tracking.

Supporting tools provide:

- search and retrieval;
- document ingestion;
- file inspection;
- sanitisation;
- indexing;
- checksums;
- policy enforcement;
- permission checks;
- command execution;
- backups;

- verification;
- audit logging.

This division allows the language model to remain relatively small.

The model does not need to internally memorise every Sentinel Linux package, every AEGIS policy, every project decision, or every device repair record. It only needs to recognise what information is required and use the retrieval system correctly.

---

5. How the Retrieval Process Works

A well-designed interaction may follow these steps.

Step 1: Understand the request

The model determines what the user is asking.

For example:

> "Which Sentinel Linux ISO phase introduced the custom installer branding?"

The system identifies important concepts such as:

- Sentinel Linux;
- ISO;
- installer;
- branding;
- phase history.

Step 2: Search the Vault

The retrieval service searches indexed records for relevant passages.

It should prefer:

- canonical records over casual notes;
- accepted releases over drafts;
- current versions over withdrawn versions;
- verified information over unverified information;
- directly relevant records over broadly similar ones.

Step 3: Rank and filter results

The system removes irrelevant, duplicate, obsolete, or unauthorised records.

It may also apply permission checks before private information is supplied to the model.

Step 4: Build a controlled context

Only the most useful records are placed into the model's working context.

This is important because excessive context can make a smaller model less reliable rather than more reliable.

Step 5: Generate the answer

The model interprets the retrieved information and answers the question.

Step 6: Verify the response

Where appropriate, the answer can be checked against:

- known versions;
- package manifests;
- checksums;
- source files;
- command output;
- policy records;
- release certificates;
- other authoritative evidence.

Step 7: Record an approved update

New information should only be stored when appropriate.

A robust system should distinguish between:

- something the user merely mentioned;
- a temporary idea;
- a draft decision;
- a confirmed decision;
- an accepted baseline;
- a superseded record;
- a withdrawn record.

This prevents the Vault from becoming a collection of contradictory statements.

---

6. How a Memory Vault Helps a Smaller Model Compete

A Memory Vault can allow a smaller model to compete surprisingly well with a larger model in several specific areas.

Personal and project continuity

A generic larger model does not automatically know the user's projects, devices, naming conventions, or previous decisions.

A smaller model connected to an accurate Vault can know:

- which release is current;
- which package was withdrawn;
- which name is official;
- which features are prohibited;
- what the user decided last week;
- where project files are stored;
- what the next development phase should be.

Within that environment, the smaller model may provide a more useful answer than a larger model with no access to the same records.

Domain specialisation

A 7B or 8B model can become highly effective when it repeatedly works within a controlled domain such as:

- a software project;
- a workshop;
- a household;
- an accounting system;
- a local network;
- a device-repair operation;
- a document archive;
- a private knowledge base.

The Vault supplies the specialised facts, while the model supplies the language interface.

Reduced reliance on internal knowledge

Without retrieval, a model may attempt to answer from its training data even when that information is incomplete or outdated.

With retrieval, the assistant can use the user's current documentation as its source of truth.

More consistent procedures

A Vault can store exact procedures for recurring operations:

1. Back up the current package.
2. verify its checksum;

3. extract it into a clean working directory;
4. run the static audit;
5. build the Debian package;
6. install it in a test environment;
7. run the acceptance gate;
8. archive the results.

A smaller model does not need to reconstruct that workflow from scratch every time.

Lower hardware requirements

Because the system relies on retrieval rather than trying to place every document into one enormous context window, it can run effectively with a smaller model and modest hardware.

Better privacy

The model, Vault, embeddings, indexes, and tools can all remain on the user's own computer.

No remote AI service is inherently required.

---

7. What a Memory Vault Cannot Do

The advantages are substantial, but the limitations must be stated honestly.

It does not increase the model's parameter count

A 7B model connected to a Vault remains a 7B model.

Its ability to understand complex relationships, perform novel reasoning, debug unfamiliar systems, and follow intricate instructions remains constrained by the underlying model.

It does not guarantee correct reasoning

The Vault may provide all the correct facts and the model may still combine them incorrectly.

Information and intelligence are related, but they are not the same thing.

It cannot fix poor retrieval automatically

The correct information may exist in the Vault but fail to appear in the search results.

This can happen because of:

- weak indexing;
- poor document chunking;
- ambiguous terminology;
- missing metadata;
- incorrect embeddings;
- unsuitable search queries;
- excessive filtering;
- permission restrictions;
- outdated indexes.

It can retrieve the wrong record

A project may contain several similar versions:

- current;
- draft;
- withdrawn;
- superseded;
- test-only;
- archived.

Without strong status metadata, the system may retrieve an old or invalid record.

Bad information remains bad information

A Vault is not automatically truthful.

If incorrect information is stored as authoritative, the assistant may reproduce it with great confidence.

This is the classic **garbage in, garbage out** problem.

Excessive context can reduce quality

Smaller models are particularly vulnerable to context overload.

Providing fifty documents when only three paragraphs are needed may lead the model to:

- ignore the most important passage;
- mix unrelated versions;
- miss a key instruction;
- produce a vague summary;
- become inconsistent;
- exceed its context capacity.

It does not make hallucinations impossible

Retrieval can reduce unsupported answers, but it cannot eliminate them.

The model may invent transitions, assumptions, explanations, file paths, command options, or conclusions that were not present in the retrieved material.

It introduces security concerns

Documents placed into a Vault may contain malicious or misleading instructions.

A secure system must treat retrieved documents as data, not as automatically trusted commands.

Sensitive records also need:

- encryption;
- access controls;
- backup protection;
- audit trails;
- clear user permissions;
- safe deletion procedures.

It does not replace a more capable model for every task

A larger model will generally remain preferable for:

- difficult unfamiliar reasoning;
- advanced software architecture;
- subtle debugging;
- complicated mathematics;
- long autonomous workflows;
- large codebase analysis;
- ambiguous instructions;
- adversarial document analysis;
- high-risk decisions.

The correct claim is therefore not that memory makes model size irrelevant.

The correct claim is:

> Memory allows a smaller model to use its limited capability much more effectively.

---

8. A Realistic Comparison

Capability	Small model alone	Small model with Vault	Larger model alone
General knowledge	Limited	Improved when relevant records exist	Usually broader
Personal memory	Minimal	Potentially excellent	Minimal without its own memory
Project continuity	Weak	Strong when records are maintained	Weak without project access
Novel reasoning	Limited	Still limited	Usually stronger
Factual local answers	Unreliable	Strong when retrieval succeeds	Unreliable without local records
Repeated procedures	Inconsistent	Highly consistent with good workflows	Generally capable
Hardware requirement	Low	Low to moderate	High
Local privacy	Easy to maintain	Easy to maintain	More expensive to maintain locally
Context management	Easier but limited	Requires careful retrieval	Usually handles more context
Cost of operation	Low	Low to moderate	Moderate to very high

A strong 7B or 8B model with excellent retrieval may feel comparable to a 14B, 32B, or occasionally larger model when answering questions about the user's own environment.

It will not consistently match the larger model across general reasoning, unfamiliar programming tasks, or broad knowledge.

---

9. Understanding AI Memory Requirements

The amount of memory required to run a model depends on several factors:

- parameter count;
- numerical precision;
- quantisation format;
- model architecture;
- context length;

- KV-cache format;
- batch size;
- number of concurrent users;
- vision or audio components;
- runtime overhead;
- whether the model is fully or partially loaded into a GPU.

Quantisation reduces model memory by storing weights at lower precision. Hugging Face describes 8-bit and 4-bit quantisation as methods for reducing memory and computational cost, allowing models to load on hardware that could not accommodate their full-precision weights.

Ollama similarly notes that quantising a model reduces memory consumption and can improve speed, although lower precision can reduce accuracy.

The tables below assume:

- local inference rather than model training;
- a dense, text-only model;
- approximately 4-bit `Q4_K_M` or comparable quantisation;
- one active user;
- a context of approximately 4,000 to 8,000 tokens;
- a modern local runtime such as llama.cpp or Ollama;
- sufficient memory reserved for the operating system.

They are planning estimates, not guarantees.

Exact requirements vary between model families.

---

10. Approximate Quantised Model Sizes

Model class	Approximate 4-bit model size	Typical role
1B–3B	0.8–2.5 GB	Basic chat, classification, simple commands
7B–8B	4.5–6 GB	General assistant, coding, retrieval, tools
12B–14B	7.5–10.5 GB	Stronger writing, coding and reasoning
20B–24B	12.5–18 GB	Advanced local assistant
27B–32B	17–24 GB	Strong reasoning and professional workloads
40B–46B	25–34 GB	High-capability workstation model
65B–70B	38–48 GB	Workstation or server-class assistant

These figures cover model weights, not the entire working memory requirement.

Additional memory is needed for:

- the KV cache;
- runtime buffers;
- prompt processing;
- output generation;
- the operating system;
- the desktop environment;
- other running applications.

A 70B 4-bit model is commonly around 40 GB or more, while an unquantised 16-bit 70B model may require roughly 140 GB just for its weights. llama.cpp discussions provide similar real-world examples of approximately 40 GB for a 4-bit 70B model and roughly 140 GB for 16-bit weights.

---

11. Running Models With System RAM Only

CPU and system-RAM inference is the most accessible configuration.

The model is loaded into normal computer memory and processed by the CPU.

Advantages

- No dedicated GPU is required.
- Large amounts of RAM are cheaper than equivalent VRAM.
- Standard desktop hardware can run surprisingly large quantised models.
- Setup is generally straightforward.
- It works well for private background assistants where immediate responses are not essential.

Disadvantages

- Token generation is substantially slower.
- Performance depends heavily on memory bandwidth.
- Older CPUs may process prompts very slowly.
- Large context windows can become frustrating.
- The CPU may remain heavily loaded.
- Laptop cooling and power limits can reduce sustained performance.

RAM-only planning guide

Model size	Minimum system RAM	Comfortable system RAM	Suggested CPU class
1B–3B	8 GB	16 GB	Modern 4-core or better
7B–8B	16 GB	24–32 GB	Modern 6–8 core

Model size	Minimum system RAM	Comfortable system RAM	Suggested CPU class
12B–14B	24 GB	32–48 GB	Modern 8–12 core
20B–24B	32 GB	48–64 GB	High-bandwidth 8–16 core
27B–32B	48 GB	64–96 GB	Modern 12–16 core
40B–46B	64 GB	96–128 GB	Workstation-class CPU
65B–70B	64 GB, very tight	96–128 GB	High-bandwidth workstation CPU

“Minimum” means that a suitable 4-bit model may load and operate.

It does not necessarily mean that the experience will be fast or pleasant.

For CPU inference, memory bandwidth often matters more than a very high core count. A modern CPU with fast dual-channel or multi-channel memory may outperform a CPU with more cores but slower memory.

Efficiency recommendation

For reliable operation, the system should retain at least:

- 4–6 GB for a lightweight operating system;
- 8 GB or more for a full desktop and normal applications;
- additional headroom for context, indexing, tools, databases, and document processing.

A system should not be planned so that the model consumes every available gigabyte.

Using swap space or disk as an extension of model memory may technically prevent a crash, but performance can become extremely poor.

---

12. Running Models Primarily in VRAM

A dedicated GPU stores model weights in video memory and performs the main calculations.

This is generally the fastest local inference configuration.

GPUs are designed to process many mathematical operations in parallel, making them well suited to machine-learning workloads.

Advantages

- Much faster prompt processing.
- Faster token generation.
- Better handling of larger contexts.

- Lower CPU load.
- Better responsiveness for interactive assistants.
- Better support for concurrent requests.

Disadvantages

- VRAM is expensive.
- Consumer graphics cards are often limited to 8–24 GB.
- Large models may require professional GPUs or multiple GPUs.
- Driver and runtime compatibility must be considered.
- The computer still requires adequate system RAM.
- Cooling and power requirements can be substantial.

Full-GPU-offload planning guide

Model size	Tight minimum VRAM	Comfortable VRAM	Recommended host RAM
1B–3B	4 GB	6–8 GB	16 GB
7B–8B	6–8 GB	10–12 GB	16–32 GB
12B–14B	12 GB	16–24 GB	32 GB
20B–24B	20 GB	24–32 GB	32–64 GB
27B–32B	24 GB	32–48 GB	32–64 GB
40B–46B	32–40 GB	48–64 GB	64 GB
65B–70B	48 GB, context-limited	64–80 GB	64–128 GB

The “tight minimum” column assumes:

- strong quantisation;
- a modest context;
- limited concurrency;
- little spare VRAM;
- a model architecture with ordinary memory behaviour.

The comfortable figure leaves room for context, runtime buffers, and practical use.

A 70B 4-bit model may fit into approximately 48 GB of VRAM under constrained settings, but 64–80 GB is a more realistic target for an efficient full-GPU experience.

---

13. Running Models With Combined RAM and VRAM

Hybrid inference divides the model between the GPU and system memory.

Some layers are placed in VRAM and the remaining layers are processed from system RAM.

Ollama reports these arrangements as a CPU/GPU split, while llama.cpp supports selecting how many layers are offloaded to the GPU.

Advantages

- Allows models larger than the GPU's VRAM capacity to run.
- Uses an existing consumer GPU rather than requiring a professional card.
- Can provide a substantial improvement over CPU-only inference.
- Offers a practical upgrade path for ordinary workstations.

Disadvantages

- It is slower than keeping the complete model in VRAM.
- The CPU portion can become a bottleneck.
- Data transfers across PCI Express add latency.
- RAM and VRAM do not always combine as one perfectly efficient memory pool.
- Some memory may be duplicated or reserved by the runtime.
- Performance can fall sharply when too much work spills back to the CPU.

Hugging Face's large-model tools use a similar priority: load as much as possible on the fastest device, then place remaining weights in CPU memory, and finally use disk only as the slowest fallback.

Practical hybrid recommendations

Model size	Practical VRAM target	Practical system RAM target	Expected result
1B–3B	4 GB	8–16 GB	Usually full or nearly full GPU operation
7B–8B	6–8 GB	16–24 GB	Very good interactive performance
12B–14B	8–12 GB	24–32 GB	Good hybrid performance
20B–24B	12–16 GB	32–48 GB	Usable, with meaningful CPU involvement
27B–32B	16–24 GB	48–64 GB	Strong but slower hybrid workstation

Model size	Practical VRAM target	Practical system RAM target	Expected result
40B–46B	24–32 GB	64–96 GB	Usable high-end hybrid system
65B–70B	32–48 GB	96–128 GB	Practical hybrid server or workstation

A 70B model can run with less than 32 GB of VRAM if sufficient system RAM is available, but it should not be described as an efficient configuration.

For a responsive hybrid system, a useful goal is to place at least half—and preferably most—of the model weights on the GPU.

The exact percentage depends on:

- GPU architecture;
- CPU performance;
- memory bandwidth;
- PCI Express speed;
- model architecture;
- context length;
- runtime implementation.

---

14. RAM, VRAM and Hybrid Inference Compared

Configuration	Capacity cost	Typical speed	Best use
RAM only	Lowest	Slowest	Background assistant, low-cost server
VRAM only	Highest	Fastest	Interactive assistant, coding, agents
RAM + VRAM	Moderate	Between the two	Consumer workstations running larger models
Disk offload	Low capacity cost	Extremely slow	Emergency loading or experimentation

RAM only

Best when:

- the budget is limited;
- privacy matters more than speed;
- the assistant runs background tasks;
- response time is not critical;
- large amounts of ordinary RAM are available.

VRAM only

Best when:

- responsiveness matters;
- the assistant is used interactively;
- large prompts are common;
- coding and tool use are important;
- several users or tasks may run concurrently.

Combined RAM and VRAM

Best when:

- the preferred model is slightly too large for the GPU;
- the user already owns a capable graphics card;
- replacing the GPU would be too expensive;
- some reduction in speed is acceptable.

Disk offload

Disk should not be treated as a substitute for RAM.

Even a fast NVMe drive is dramatically slower than RAM or VRAM for repeated model operations. It may allow a model to start, but the resulting performance is often unsuitable for normal interactive use.

---

15. Context Length Also Consumes Memory

Model size is only one part of the calculation.

The context window holds:

- the system instructions;
- the user's prompt;
- retrieved Vault records;
- conversation history;
- tool results;
- the model's generated response.

During inference, the runtime maintains a **KV cache** representing previous tokens. The memory required by this cache generally increases with context length.

llama.cpp's multi-GPU guidance notes that reducing context size reduces KV-cache demand, and that additional concurrent sequences require additional cache allocations.

Ollama also adjusts default context sizes based on available VRAM and advises avoiding CPU offload where maximum performance is required.

A model that fits comfortably at 4,000 tokens may exceed available memory at:

- 32,000 tokens;
- 64,000 tokens;
- 128,000 tokens;
- multiple simultaneous sessions.

This is one reason a Memory Vault should retrieve a small amount of highly relevant information rather than inserting the entire Vault into every prompt.

Good retrieval is not only more accurate. It is more memory-efficient.

---

16. Unified Memory Systems

Some computers use unified memory shared by the CPU and GPU.

This is common on modern system-on-chip designs and integrated graphics systems.

Unified memory can reduce the need to copy data between separate RAM and VRAM pools. It may allow a GPU to access a much larger shared memory allocation than a conventional consumer graphics card.

However, shared memory capacity should not automatically be treated as equivalent to dedicated high-bandwidth VRAM.

Performance depends on:

- memory bandwidth;
- chip architecture;
- runtime support;
- memory reserved by the operating system;
- thermal limits;
- how effectively the model uses the GPU.

For example, a computer with 64 GB of unified memory does not have all 64 GB available exclusively to the model. The operating system, applications, display system, context cache, and runtime also need memory.

For comfortable 70B operation on a unified-memory machine, 96–128 GB is much safer than attempting to use a 64 GB system at its absolute limit.

---

17. Model Recommendations by Hardware Class

8 GB system RAM, no dedicated GPU

Recommended:

- 1B–3B models;
- short contexts;
- basic chat;
- command classification;
- simple document retrieval.

Not recommended:

- 7B models while running a full desktop;
- large document contexts;
- multiple models at once.

16 GB system RAM, no dedicated GPU

Recommended:

- 3B models comfortably;
- 7B–8B models with controlled context;
- lightweight Memory Vault retrieval;
- simple local assistant workloads.

Expected limitation:

- CPU generation may be slow.

24 GB system RAM, no dedicated GPU

Recommended:

- 7B–8B as the practical default;
- 12B–14B in 4-bit form for experimentation;
- one model loaded at a time;
- carefully limited context.

This is a strong configuration for an AEGIS-style 7B assistant.

A 14B model may run, but an 8B model may provide a better overall experience because it leaves more memory for the Vault, desktop, tools, and indexing services.

32 GB RAM with 8–12 GB VRAM

Recommended:

- 7B–8B with excellent GPU acceleration;
- 12B–14B with full or partial GPU offload;
- substantial retrieval and tool use;
- moderate context sizes.

This is one of the best value configurations for a responsive private assistant.

64 GB RAM with 16–24 GB VRAM

Recommended:

- 14B models comfortably;
- 20B–24B models;
- 27B–32B through hybrid inference;
- larger context windows;
- stronger coding and document analysis.

This is a capable local AI workstation.

96–128 GB RAM with 32–48 GB VRAM

Recommended:

- 32B models with strong acceleration;
- 40B–46B hybrid inference;
- 70B hybrid inference;
- large Vaults;
- multi-tool agents;
- more concurrent workloads.

64–80 GB dedicated VRAM

Recommended:

- efficient full-GPU 70B inference;
- long contexts, subject to model architecture;
- professional local assistant workloads;
- multiple sessions;
- advanced coding and analysis.

---

18. Why the Best Model Is Not Always the Largest Model

A larger model that consumes nearly every available resource may provide a worse practical assistant experience than a smaller model with sufficient headroom.

An overloaded system may suffer from:

- slow model startup;
- memory pressure;
- swapping;
- desktop freezing;
- reduced context capacity;
- thermal throttling;
- crashes;
- inability to run retrieval tools;
- inability to index documents;
- poor multitasking.

The AI model is only one component of the complete assistant.

An AEGIS-style system may also need memory for:

- the operating system;
- the user interface;
- Ollama or another runtime;
- the Memory Vault database;
- embedding generation;
- document parsing;
- search indexes;
- security tools;
- browser integration;
- command execution;
- logs;
- backups;
- file sanitisation;
- application integrations.

A smaller model that leaves room for these services may produce a much more useful complete system.

---

19. Designing the Vault for Smaller Models

Smaller models benefit from clear, compact, authoritative records.

Prefer structured canonical records

A record should identify:

- subject;
- status;
- current value;
- source;
- verification date;
- superseded records;
- relevant permissions.

Example:

```
Project: Sentinel Linux ISO
Release: Sentinel Linux 1.0 AMBROSO
Status: Phase 6 accepted and locked
Canonical ISO:
  /srv/sentinel-linux-iso/out/Sentinel-Linux-v1.0-AMBROSO-Phase6-Installer-
amd64.iso
Withdrawn baseline:
  Phase 6 v1.0
Pending requirement:
  Exclude mate-calc during the next corrective phase
Last verified:
  2026-07-25
```

This is easier for a small model to interpret than hundreds of pages of mixed conversation.

Separate current truth from history

The Vault should distinguish:

- canonical;
- accepted;
- locked;
- draft;
- experimental;
- withdrawn;
- superseded;
- archived.

Historical information remains useful, but it must not be mistaken for the current state.

Store decisions with reasons

A decision record should explain:

- what was decided;
- why it was decided;
- who approved it;
- when it became active;
- what it replaced.

Retrieve narrowly

The retrieval system should provide the smallest set of records needed to answer the question.

Preserve provenance

The model should be able to identify where a fact came from.

Verify important answers

For package versions, checksums, commands, financial records, permissions, or security-sensitive operations, the system should verify the answer against an authoritative source.

Use tools rather than language alone

A small model should not be expected to calculate, inspect packages, compare files, or validate checksums purely through text prediction.

Where possible, it should call a deterministic tool.

---

20. The Most Effective Local-AI Formula

For many users, the strongest practical architecture is not:

> The largest model the computer can barely load.

It is:

> The most capable model the computer can run comfortably, combined with accurate memory, strong retrieval, useful tools, and verification.

A practical private assistant may therefore consist of:

- a modern 7B–14B model;
- 4-bit quantisation;
- a structured Memory Vault;
- semantic and keyword search;
- canonical project records;
- permission controls;
- source tracking;
- deterministic tools;
- document sanitisation;
- response verification;
- sufficient unused memory for normal system operation.

This configuration can outperform a much larger unsupported model in everyday usefulness.

It starts faster, runs more reliably, consumes less power, leaves room for applications, and can be specialised around the user's real work.

Conclusion

A Memory Vault does not make model size meaningless.

It does not convert a 7B model into a 70B model, create missing reasoning ability, or guarantee correct answers.

What it does is solve a different and equally important problem.

It gives the assistant persistent access to the information that matters:

- the user's projects;
- the user's decisions;
- the user's documents;
- the user's procedures;
- the user's environment;
- the user's history;
- the user's rules.

A smaller model with no memory must repeatedly guess.

A smaller model with a well-maintained Vault can retrieve, reference, and act on verified information.

Within a familiar domain, that combination can compete with—and sometimes outperform—a larger model that lacks the same context.

The honest design principle is therefore:

> Use model size to provide enough reasoning capability. Use the Memory Vault to provide continuity, specialisation, and factual grounding. Use tools and verification to compensate for the weaknesses of language generation.

For a private system such as AEGIS, this creates a sensible path forward.

A capable 7B or 14B model can run on ordinary hardware. The Vault can preserve years of knowledge outside the model. Retrieval can supply only what is relevant. Tools can perform exact operations. Verification can reduce unsupported conclusions. The complete system can remain local, private, maintainable, and under the user's control.

The result is not an artificial intelligence that knows everything.

It is something more practical:

an assistant that reliably knows the information that matters to its user.