Agentic Parallel Book Writing:
Hierarchical Multi-Agent Coordination of Local Language Models for Coherent Long-Form Fiction and Instructional Publishing

Shyamal S. Chandra
Domino Data Systems
shyamal@dominodata.com

July 2026

Abstract

We present *Agentic Parallel Book Writer*, a Rust orchestration system that generates book-length manuscripts—fiction novels, instructional textbooks, university hardbacks, language courses, and media-rich STEM texts—entirely on local hardware using pools of Ollama-served language models. The core insight is **canon-first hierarchical decomposition**: a Director agent freezes global structure (synopsis, curriculum, or language scope) before parallel Writer agents draft scenes or sections, while a shared JSON *canon* serves as source of truth and crash-safe checkpoint. We describe four- and six-tier pipelines for fiction and standard textbooks, a seven-tier extension with a Media stage (Ollama image generation, ManimRedux animated diagrams, and text-to-video placeholders), tier-specific concurrency caps that avoid KV-cache failures on Apple Silicon, and native LaTeX assembly with optional XeLaTeX for Unicode language courses. On an M2 Ultra with `llama3.1:8b`, 90 000-word fiction drafts complete in 1–2 hours wall-clock with 100% completion after retry logic. The system demonstrates that structured multi-agent coordination, not larger context windows alone, scales local LLMs to novel-length coherent output without third-party APIs.

1 Introduction

Long-form book generation with language models fails in two predictable ways: **plot drift** when chapters are drafted without a frozen ending, and **throughput collapse** when every agent tier hammers a single inference server. Commercial APIs sidestep the second problem but leak unpublished IP. We target authors, educators, and researchers who need book-length output on air-gapped or privacy-sensitive hardware.

Our contributions are:

1. A **shared canon model** with atomic checkpointing and scene-granular resume.
2. **Tier-specific concurrency**: Writer slots at pool capacity; Reviewer at $\frac{1}{4}$ Writer concurrency to avoid llama.cpp KV failures.
3. **Seven publishing modes** from fiction through media-rich textbooks with automated figure and animation pipelines.
4. **Native LaTeX emission** (no Markdown intermediate) with mode-specific packages (`graphicx`, `media9`, hardback 7×10 in layout).
5. Measured wall-clock benchmarks on Apple Silicon with permissive JSON parsing for small-model reliability.

2 System architecture

2.1 Shared canon

All agents read and write a single `canon.json` containing title, outline, style packet, per-chapter plans, scene/section drafts, exercise sets, and stage metadata. Checkpoints use write-then-rename for crash safety. Resume reloads the canon and skips completed scenes via draft keys.

2.2 Agent tiers

Fiction (4 tiers). Director → Planner → Writer → Reviewer. The Director writes synopsis *including the ending* before any scene prose. Writers receive style packet + previous scene tail; Reviewers polish whole chapters.

Instructional (6 tiers). Director → Planner → Writer → Reviewer → Exercises → Answers. Defaults: 15 chapters, 4 sections/chapter, 3500 words/chapter, 10 problems/chapter.

Media-rich textbook (7 tiers). Same as instructional plus **Media** after Reviewer:

- `textbook-pictures`: Ollama `x/z-image-turbo` via `/v1/images/generations`; figures sprayed into prose at pseudo-random paragraph boundaries.
- `textbook-video`: ffmpeg title-card MP4 placeholders at `\includemedia` paths; stub for future text-to-video.
- `textbook-animated`: ManimRedux (Rust Manim port) renders diagram AVI, ffmpeg transcodes to MP4, embedded with `media9`.

2.3 Parallel Ollama pool

By default four `ollama serve` processes bind ports 11500–11503 with `OLLAMA_NUM_PARALLEL=4` each (16 concurrent writers). External endpoints are supported for existing clusters. Request timeouts default to 900 s per call; `num_ctx=8192`.

3 LaTeX assembly

Agents emit fragment LaTeX only (`\section{}`, no `\documentclass`). Assembly strips code fences and forbidden commands, concatenates chapters, appends exercise and answer appendices for textbooks, and invokes `pdflatex` (or `xelatex` for Unicode language books). Media modes create `figures/` and `media/` asset directories with README stubs.

4 Reliability engineering

Small models often wrap JSON in markdown fences. We extract the first balanced JSON object/array and apply deterministic fallbacks for planner/reviewer failures. Before tier-specific caps, reviewer-stage KV-cache errors aborted ∼24% of 90k-word runs; with Reviewer concurrency at $\frac{1}{4}$ Writer slots and transient retries, 50 trials reached 100% completion.

Scene summaries (last 2–3 sentences) inject into the next Writer prompt so parallel scenes maintain narrative glue despite out-of-order completion.

5 Evaluation

Table 1 summarizes fiction pipeline timings on M2 Ultra, four servers, sixteen writers, Q6_K `llama3.1:8b`.

Table 1: Fiction pipeline wall-clock (M2 Ultra, 16-way writers)

Target	Writer	Reviewer	Total
90k words	45–90 min	10–20 min	1–2 h
150k words	95–130 min	14–22 min	110–155 min
300k words	190–245 min	28–45 min	220–295 min

Director planning completes in 25–45 s; Planner (20 chapters, 16-way) ∼1 min. Assembly + `pdflatex`: seconds unless Unicode XeLaTeX pass.

6 Media pipeline details

Pictures. An LLM plans figure captions and prompts per section; images decode from base64 API responses into `figures/chNN_fig_SS_slug.png`. Injection uses section-aware pseudo-random paragraph slots (`-figures-per-section`, default 2).

Animated. `manim-scene-runner` (optional `-features manim-redux`) renders concept/process/compare scenes to AVI; ffmpeg produces H.264 MP4. Missing runner writes stub AVI with warning.

Video. Placeholder MP4s document future text-to-video integration; agents already emit `\includemedia` blocks for Adobe Reader playback.

7 Related work

Hierarchical LLM writing appears in recursive summarization and outline-then-expand systems; our distinction is *parallel section writers with canon*

re-injection and *production LaTeX* for print and digital distribution. Local inference orchestration builds on Ollama's OpenAI-compatible API and llama.cpp parallelism.

8 Conclusion

Structured decomposition with a frozen canon enables local 7–8B models to draft book-length manuscripts in hours, not days. Media tiers extend the same orchestration to illustrated and animated STEM publishing without cloud image or video APIs. Future work includes tier-specific model routing (large Director, small Writer), automated quality scoring, and wiring a text-to-video backend for `textbook-video` mode.

Availability

Public overview and this paper: https://domino-data-systems.github.io/agentic-parallel-book-writer/. Implementation source is maintained separately; CLI reference and media pipeline docs ship with the repository under `docs/`.