AI-Assisted Code Review

Alexey Rybalchenko
Software Development for Experiments (SDE) group,
GSI Helmholtz Centre for Heavy Ion Research

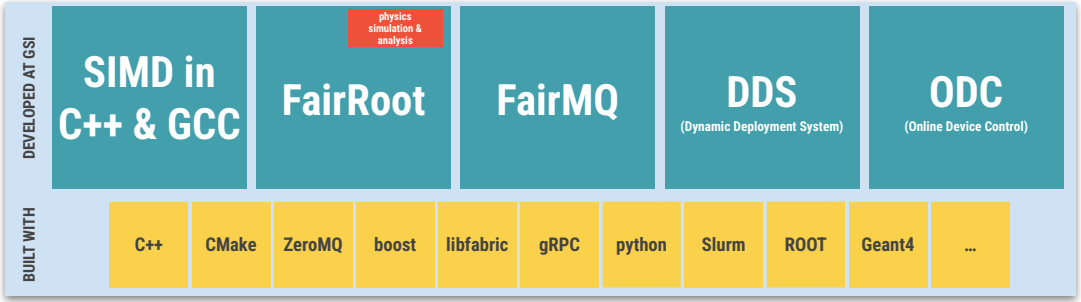
11th Annual MT Meeting
GSI, November 5, 2025

AI-Assisted Code Review

in the GSI SDE group

Software Development for Experiments (SDE) Group

The SDE group (7 people) develops and maintains common scientific software for the physics experiments in close collaboration with the experiment groups and High Energy and Nuclear Physics community.



https://github.com/FairRootGroup

https://github.com/GSI-HPC

- Using **CodeRabbit AI** code review (LLM-based) since ~1.5 years to assist with code review on several production projects.
- In parallel, since ~1 year, developing an open source LLM code review tool **Pearbot** for use with open-weights models & to gain experience with LLM tooling.

CodeRabbit

Overview

https://coderabbit.ai/

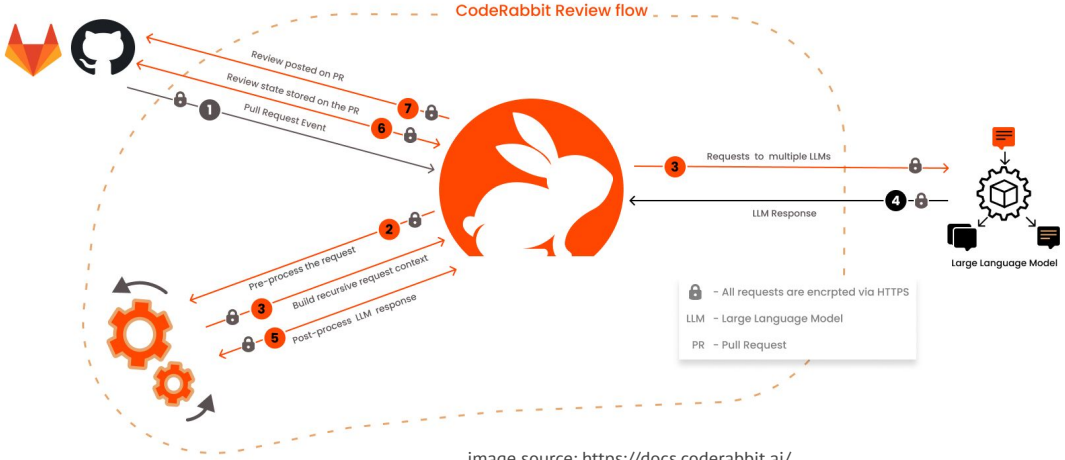
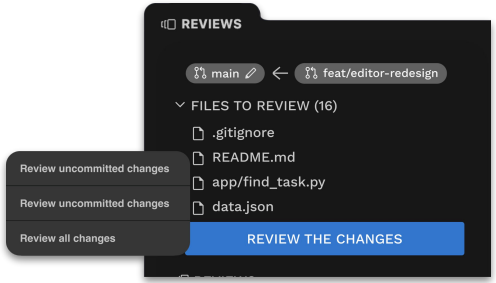


image source: https://docs.coderabbit.ai/

- LLM-based Pull Request reviewer via a Github/Gitlab App.
- Previously open source, now a closed project.
- Free to use for open source projects.
- Combination of commercial LLMs.

- New (Since May 2025): VS Code plugin to provide reviews before code goes to repository.

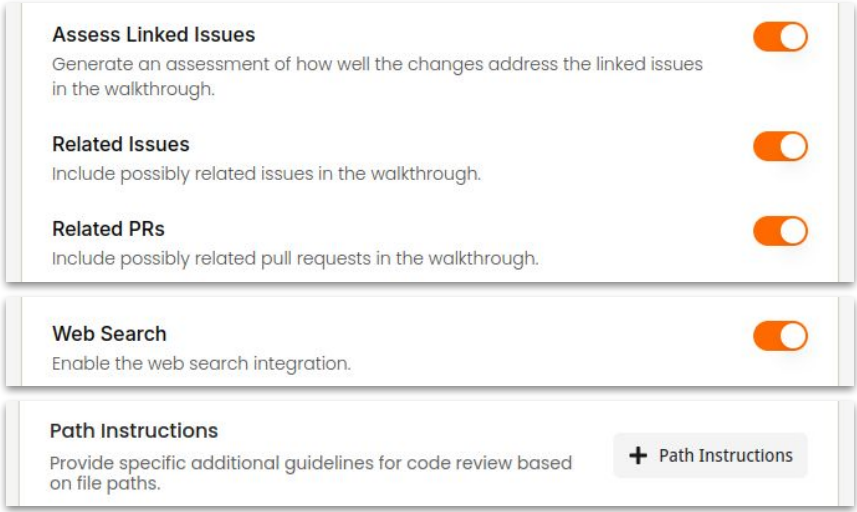


CodeRabbit

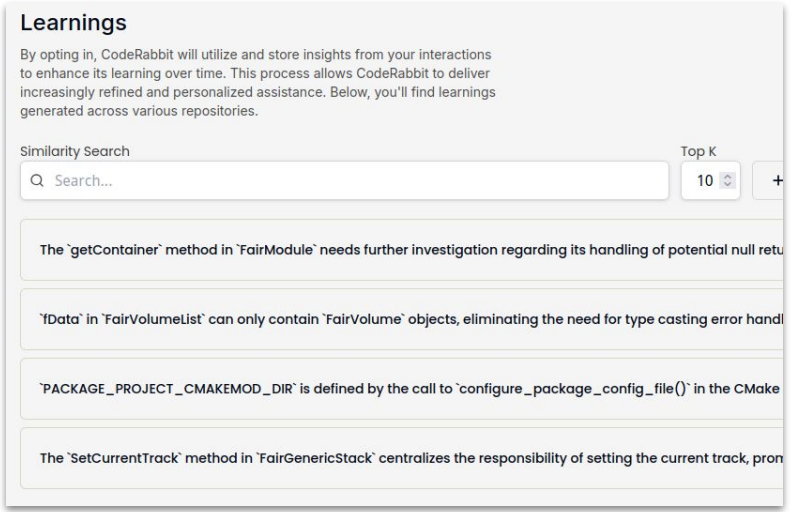
Pre-processing & post-processing

Review quality can be significantly improved by providing additional contextual information, relevant to the submitted changes. Among obvious things are PR text, commit messages, comments, relevant coding guidelines, parts of relevant code base, etc.

Details about what exactly is used and how are not revealed by CodeRabbit.



and more…

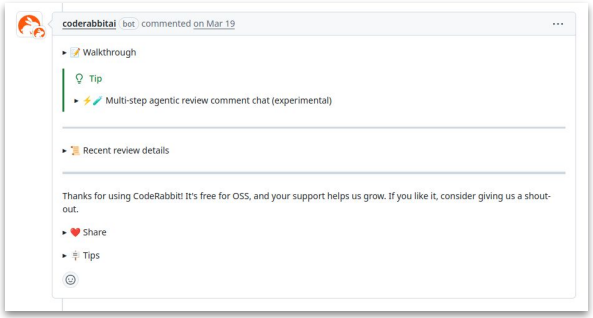


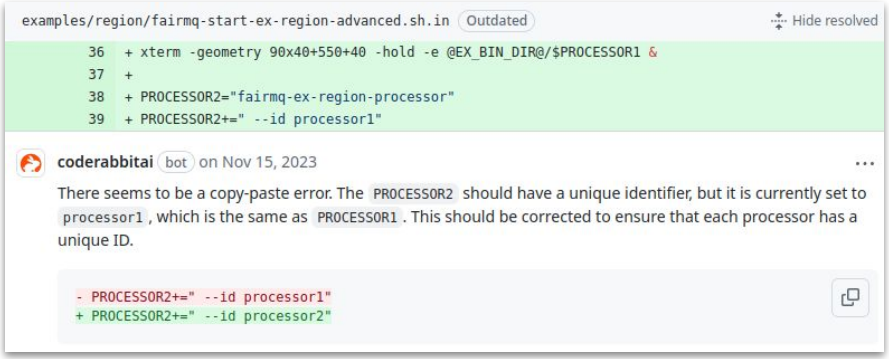
CodeRabbit

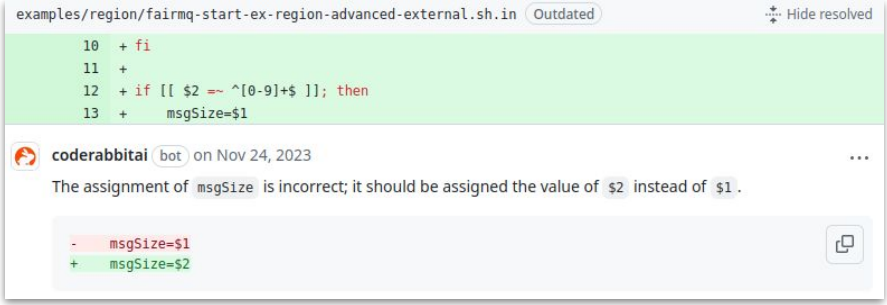
… and LLM-based review in general. Our experience

+ Good at identifying logical errors, that other automated tools or even human review overlook.
+ Quick 24/7 feedback, unfamiliar with project biases, scalable, multi-lingual, customizable, with a broad knowledge base, can be kept up-to date with new data.
+ Can learn from: related issues, related PRs, previous interactions, web search.

- May produce unnecessary output, when no actionable changes are necessary or such are not deduced by the model.
- Limited understanding of complex projects (can be improved with context engineering).
- Potential for false positives, flagging issues that aren't actually problematic.
- Commercial projects may include some promotional output:







CodeRabbit

Tool use & nitpicks

CHANGELOG.md

```
+
+ ### Breaking Changes
+ * Event display
+   * Removed     void FairEventManager::LoadXMLDetector(TGeoNode* node, FairXMLNode* xml,
  Int_t depth = 0)
```

coderabbitai bot on Jul 22, 2024

Fix markdownlint issue.

Remove spaces inside emphasis markers.

```
-  * Removed     void FairEventManager::LoadXMLDetector(TGeoNode* node, FairXMLNode* xml, Int_t
+  * Removed void FairEventManager::LoadXMLDetector(TGeoNode* node, FairXMLNode* xml, Int_t depth = 0)
```

▾ Committable suggestion

‼ IMPORTANT

Carefully review the code before committing. Ensure that it accurately replaces the highlighted code, contains no missing lines, and has no issues with indentation. Thoroughly test & benchmark the code to ensure it meets the requirements.

Suggested change

```
11 -   * Removed     void FairEventManager::LoadXMLDetector(TGeoNode* node, FairXMLNode* xml,
   Int_t depth = 0)
11 +   * Removed void FairEventManager::LoadXMLDetector(TGeoNode* node, FairXMLNode* xml, Int_t
   depth = 0)
```

Commit suggestion ▾ | Add suggestion to batch

▾ Tools

▾ Markdownlint

11-11: null

Spaces inside emphasis markers

(MD037, no-space-in-emphasis)

coderabbitai bot left a comment

Actionable comments posted: 0

▾ 🧹 Nitpick comments (1)

▾ fairroot/base/sim/FairMCApplication.h (1)

`222-226` : **Fix typo and add const qualifier for better design.**

The getter implementation is correct and follows good naming conventions. However, there are two minor improvements:

1. Fix the typo "goint" → "going" in the documentation
2. Add `const` qualifier since the method doesn't modify object state

Apply this diff to improve the method:

```
-    /**
-     * Returns the state of the flag deciding whether the current event
-     * is goint to be stored.
-     */
-    Bool_t GetSaveCurrentEvent() { return fSaveCurrentEvent; }
+    /**
+     * Returns the state of the flag deciding whether the current event
+     * is going to be stored.
+     */
+    Bool_t GetSaveCurrentEvent() const { return fSaveCurrentEvent; }
```

The addition successfully addresses the PR objective of providing read access to the `fSaveCurrentEvent` flag and complements the existing setter nicely.

▾ 📜 Review details

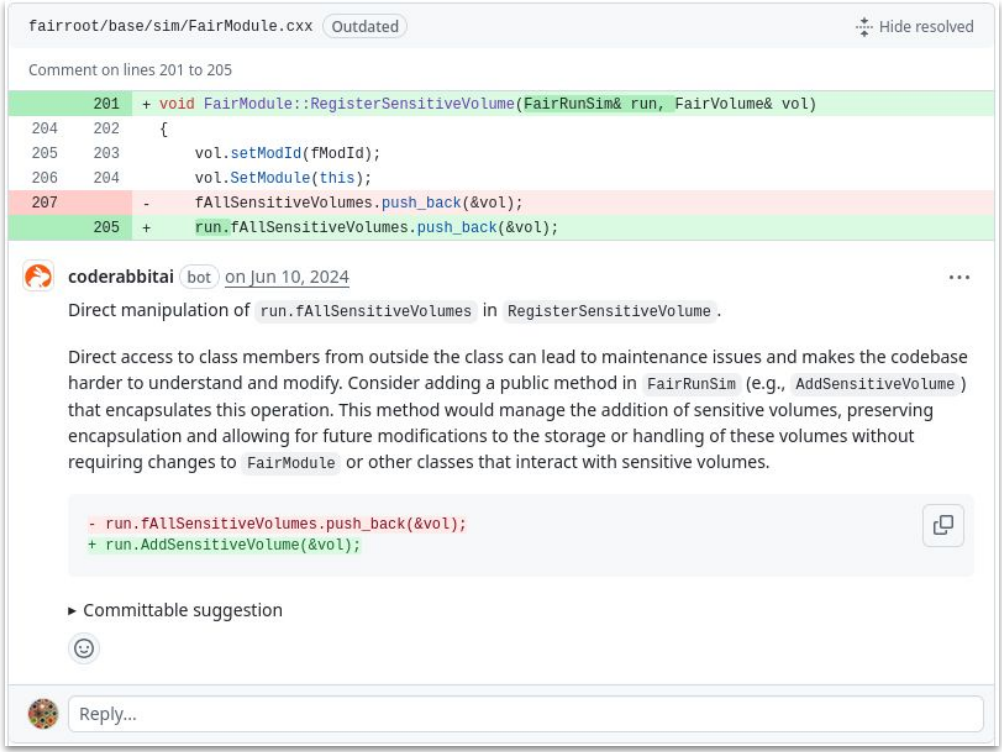
Configuration used: CodeRabbit UI
Review profile: CHILL
Plan: Pro

▸ 📥 Commits

▸ 📒 Files selected for processing (1)

CodeRabbit

Actionable?



fairroot/base/sim/FairModule.cxx Outdated — Hide resolved

Comment on lines 201 to 205

```
201 + void FairModule::RegisterSensitiveVolume(FairRunSim& run, FairVolume& vol)
202   {
203       vol.setModId(fModId);
204       vol.SetModule(this);
207 -     fAllSensitiveVolumes.push_back(&vol);
205 +     run.fAllSensitiveVolumes.push_back(&vol);
```

coderabbitai bot on Jun 10, 2024

Direct manipulation of `run.fAllSensitiveVolumes` in `RegisterSensitiveVolume`.

Direct access to class members from outside the class can lead to maintenance issues and makes the codebase harder to understand and modify. Consider adding a public method in `FairRunSim` (e.g., `AddSensitiveVolume`) that encapsulates this operation. This method would manage the addition of sensitive volumes, preserving encapsulation and allowing for future modifications to the storage or handling of these volumes without requiring changes to `FairModule` or other classes that interact with sensitive volumes.

```
- run.fAllSensitiveVolumes.push_back(&vol);
+ run.AddSensitiveVolume(&vol);
```

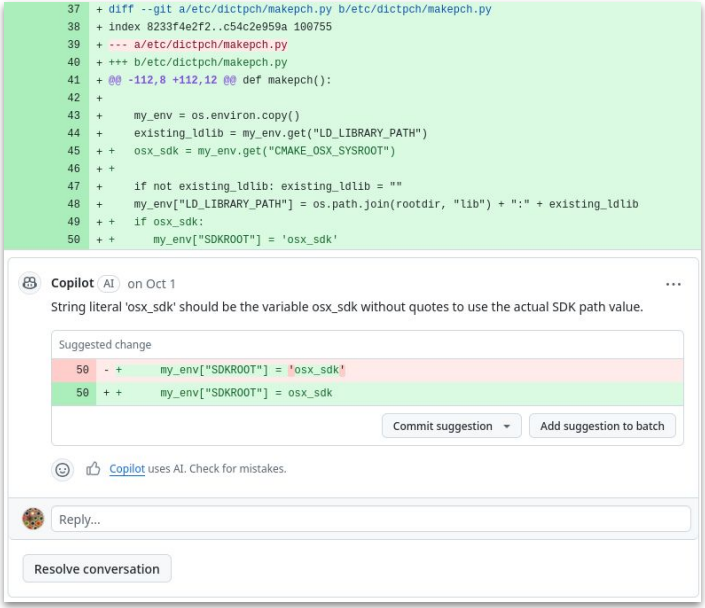
▸ Committable suggestion

Reply...

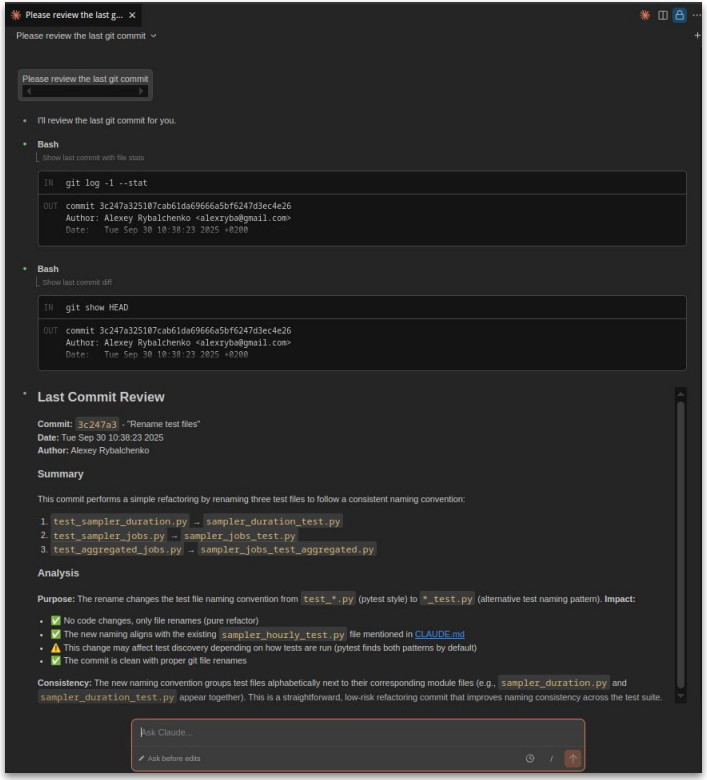
AI Code Review

Other Tools

GitHub Copilot



🌟 Local review with Claude Code



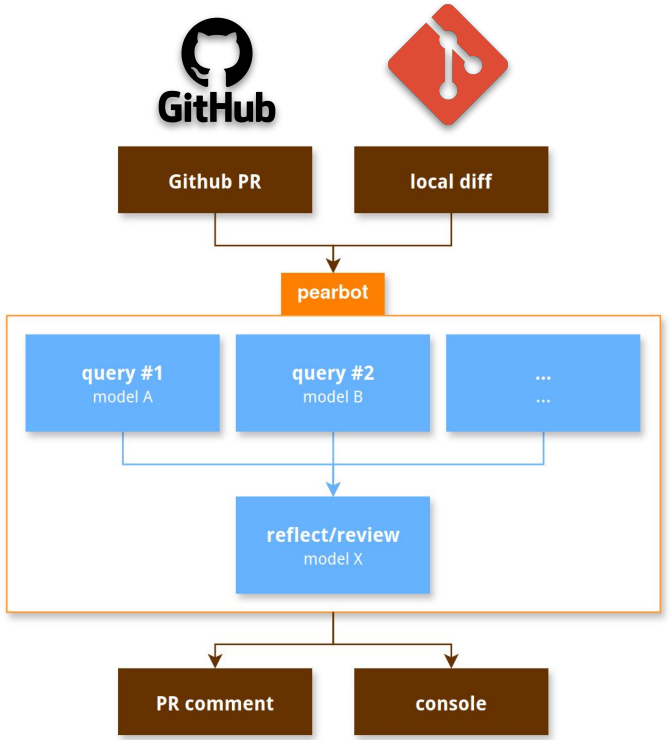
https://kodus.io/	open source	
https://www.greptile.com	closed source	
https://www.getpanto.ai/	closed source	
https://docs.claude.com/en/docs/claude-code/quickstart	commercial	

Pearbot

Overview

https://github.com/GSI-HPC/pearbot

- GitHub App for reviewing Pull Requests.
- Local execution mode for diffs or annotated commits.
- LLM ensemble approach for improved results.
- Execution on low-end hardware and/or without GPU.
- Customizable model(s) & prompt(s).



Pearbot

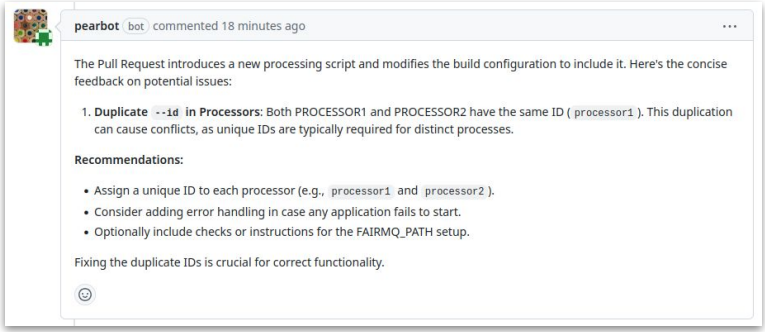
Usage

As a GitHub App:
`python pearbot.py --server`

Analyze a local diff file:
`python pearbot.py --diff path/to/your/diff/file`

Or pipe a diff directly:
`git diff | python pearbot.py`

Generate detailed output with commit messages, e.g.:
`git format-patch HEAD~3..HEAD --stdout | python pearbot.py`

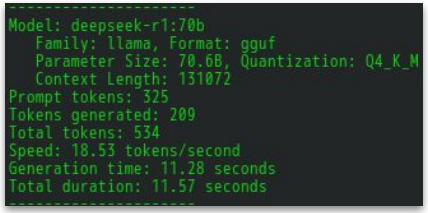


Pearbot

Backend

ollama: open-source LLM server, written in Go, backed by **llama.cpp** (C++)

- Efficient serving of large language models
- CPU/GPU/CPU+GPU hybrid inference to partially accelerate models larger than the total VRAM capacity
- Supports many model architectures:
  deepseek2, llama, gemma2, qwen2, …
- Support for multitude of model quantization techniques for faster inference and reduced memory use
- Usage Metrics



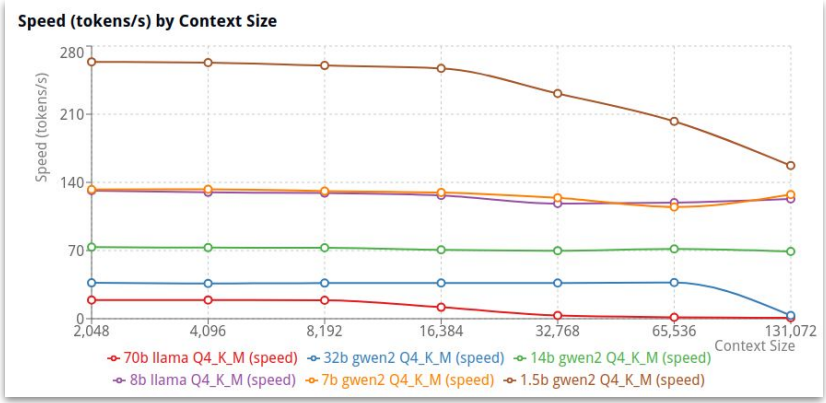
ollama has a very minimal feature set when it comes to things like:

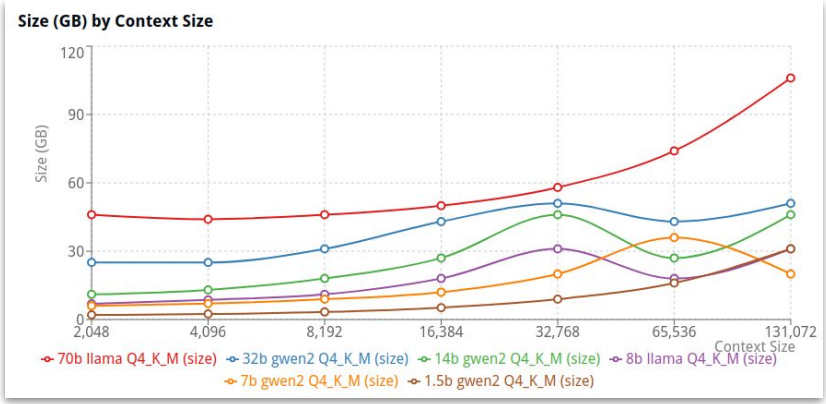
- queue visibility
- additional info (how many tokens does this prompt take?)
- multi-server setups with “big” GPUs

Alternatives for HPC: e.g. **vLLM**, SGLang, LMDeploy, TensorRT-LLM

ollama

context size performance impact





Common gotcha with ollama is the default context size setting of 2048 tokens.

Comparing the performance of different distillations of DeepSeek-R1 with varying context size:

- generation speed
- model size in memory
- GPU usage

On Nvidia RTX 6000 Ada Generation, 48GB VRAM

Matrix View: GPU Usage (%)

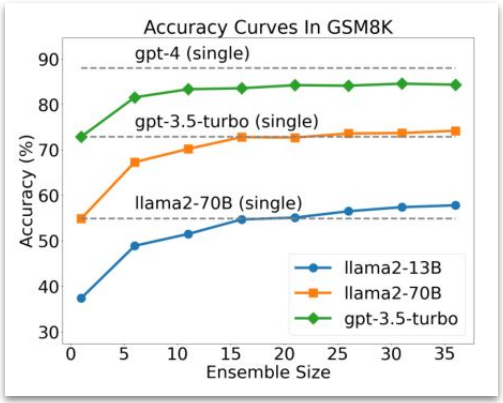
Model	2,048	4,096	8,192	16,384	32,768	65,536	131,072
70b llama	100	100	100	98	86	67	47
32b gwen2	100	100	100	100	100	100	72
14b gwen2	100	100	100	100	100	100	100
8b llama	100	100	100	100	100	100	100
7b gwen2	100	100	100	100	100	100	100
1.5b gwen2	100	100	100	100	100	100	100

Pearbot

Quality improvements over the base model

1. **Multi-Model Initial Reviews:**
   - Multiple LLMs (ensemble) generate initial code reviews
     - "generate independent thoughts" that may touch different aspects of the problem.
2. **Reflection[1][2] by a "decider" model:**
   - A separate, potentially more advanced model analyzes the initial reviews.
   - Refines the generated feedback, rejects potentially less impactful comments.
   - It prioritizes the most important issues and suggestions (prompt-dependent).
3. **Prompt improvements:**
   - Specific & useful code review examples.
   - Examples include Chain-of-Thought[3] type of reviews, that include some reasoning why the suggestions would be good.

[1] Madaan, Aman, et al. "Self-refine: Iterative refinement with self-feedback." Advances in Neural Information Processing Systems 36 (2024). https://doi.org/10.48550/arXiv.2303.17651
[2] Shinn, Noah, et al. "Reflexion: Language agents with verbal reinforcement learning." Advances in Neural Information Processing Systems 36 (2024). https://doi.org/10.48550/arXiv.2303.11366
[3] Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." Advances in neural information processing systems 35 (2022). https://doi.org/10.48550/arXiv.2201.11903



Accuracy of multi-agent approach Grade School Math 8K problems.
image from: Li, Junyou, et al. "More agents is all you need." arXiv preprint arXiv:2402.05120 (2024).

→ Good results with smaller PRs (even with small quantized models). Focus suffers on larger PRs. Issues when hitting context size limits.

→ DeepSeek-R1's approach mostly overshadows these optimizations

Trained to generate "thinking tokens" – reflecting on the problem from different perspectives before giving a final answer.
Similar open-weights models: gpt-oss, qwen3, magistral
And most commercial models have a thinking variant

Pearbot

TODO list & general improvement ideas

- Inline comments.
- Improve handling with large PRs/commits: focus vs. context size.
- Additional context:
  - related issues
  - code history
  - experience from past interactions

  Balance additional context with keeping the focus on the task - avoid distractions!
- Rejection of useless output, e.g. for automated reviews no found issues should produce no comments, but only a green GitHub checkmark.
- Balance larger & smaller models for different tasks → cost/efficiency optimization.
- Deployment with larger models: make ollama dependency optional, use OpenAI API instead.

https://www.greptile.com/blog/make-llms-shut-up

✖ prompting

✖ LLM-as-a-judge

✅ clustering in a vector database

AI Code Review

Conclusion

- LLMs and the tools around them are improving fast.
- AI reviews are useful, with decreasing number of downsides.
- As reviewer: low or no risk.
- Tools development & experience **>** benchmarks.
- Open-weights LLMs are viable.
- Can be very resource-hungry.
- With on-premises execution, balance where possible/needed:
    1. smaller models (fewer parameters)
    2. quantization
    3. context size
    4. backend optimizations (prefix caching, Flash attention, etc.)
- Biggest challenge: providing focused context for the most helpful review.