CERFACS

CENTRE EUROPÉEN DE RECHERCHE ET DE FORMATION AVANCÉE EN CALCUL SCIENTIFIQUE

ISC High Performance

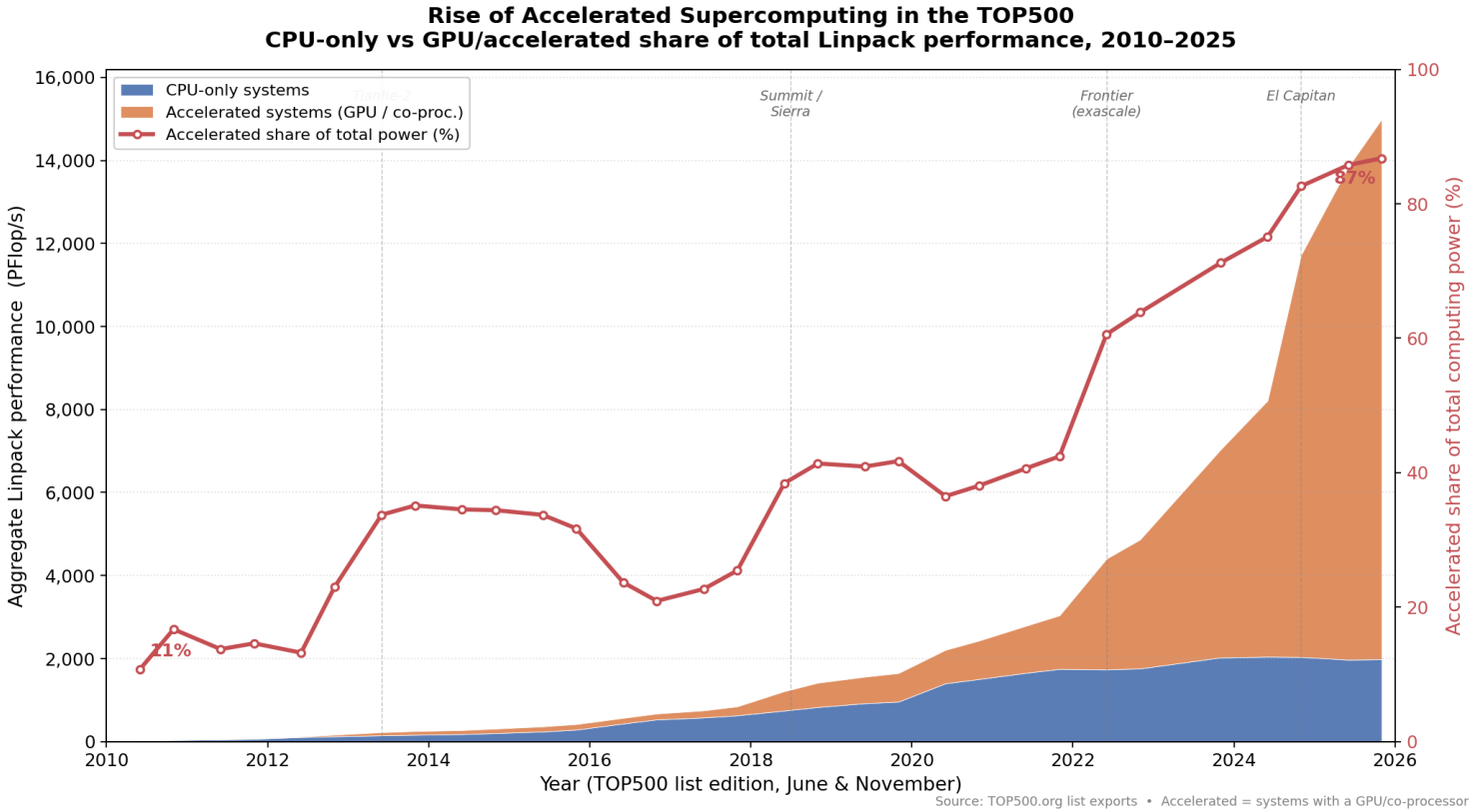
ISC 2026 · The Second International Workshop on Foundational Large Language Models Advances for HPC

LLMs Evaluation for Fortran HPC Code Generation and Translation

T. Marzlin · M. Ghenai · A. Dauptain

COOP team — CERFACS, Toulouse, France - 2026

Supercomputing has gone heterogeneous



Most TOP500 FLOPS now come from GPU-accelerated systems.

A decade-long shift, accelerated by the AI boom (since 2015).

For HPC application owners, this means one thing:

massive code modernization and porting effort.

If LLMs can write code, can they also make HPC porting easier ?

OUR CONTEXT

...but the code we have to port is Fortran

72% **of CPU cycles on SuperMUC** allocated to Fortran projects
Lohmann et al., 2017

~1.5% **share of Fortran in the** TIOBE programming index
≈ what LLMs likely saw in training

We had an in-house CFD code in Fortran, with OpenACC directives ➛ Migration to OpenMP for portability across various HPC clusters.

Brittle, manual, hard to verify.

OUR RESEARCH PROGRAM

Three steps · 1 : Evaluate

01 TODAY

Evaluate

Find out what models can actually do on Fortran HPC, raw, today.

- 45 models · 177 tasks
- FortranHumanEval + OpenACC→OpenMP
- Single-shot, on-prem

This paper.

02 NEXT PAPER

Fine-tune

Build Fortran-HPC training corpora from in-house codes; close the on-prem gap.

- Curate from scientific libraries
- Proprietary code the public models never saw
- Performance-aligned where possible

Coming next.

03 AFTER

Agents & workflow

Wrap models in compile-and-test loops, multi-agent orchestration, repo-scale workflows.

- Self-correcting pipelines
- Deterministic prompt skills
- IDE & chat integration

Paper after that.

OUR RESEARCH PROGRAM

Three steps · 2 : Fine-tune

01 TODAY

Evaluate

Find out what models can actually do on Fortran HPC, raw, today.

- 45 models · 177 tasks
- FortranHumanEval + OpenACC→OpenMP
- Single-shot, on-prem

This paper.

02 NEXT PAPER

Fine-tune

Build Fortran-HPC training corpora from in-house codes; close the on-prem gap.

- Curate from scientific libraries
- Proprietary code the public models never saw
- Performance-aligned where possible

Coming next.

03 AFTER

Agents & workflow

Wrap models in compile-and-test loops, multi-agent orchestration, repo-scale workflows.

- Self-correcting pipelines
- Deterministic prompt skills
- IDE & chat integration

Paper after that.

OUR RESEARCH PROGRAM

Three steps · 3 : Agents and workflow

01 TODAY

Evaluate

Find out what models can actually do on Fortran HPC, raw, today.

- 45 models · 177 tasks
- FortranHumanEval + OpenACC→OpenMP
- Single-shot, on-prem

This paper.

02 NEXT PAPER

Fine-tune

Build Fortran-HPC training corpora from in-house codes; close the on-prem gap.

- Curate from scientific libraries
- Proprietary code the public models never saw
- Performance-aligned where possible

Coming next.

03 AFTER

Agents & workflow

Wrap models in compile-and-test loops, multi-agent orchestration, repo-scale workflows.

- Self-correcting pipelines
- Deterministic prompt skills
- IDE & chat integration

And after that.

Three steps · today is step 1

01 TODAY

Evaluate

Find out what models can actually do on Fortran HPC, raw, today.

- 45 models · 177 tasks
- FortranHumanEval + OpenACC→OpenMP
- Single-shot, on-prem

This paper.

Fortran HPC benchmark : The only one we found was **FortranHumanEval** — a Fortran translation of the Python HumanEval suite. Not HPC-specific, but the best available foundation.

Single-shot prompting on purpose: in interactive use, if the first answer isn't right, PhD students won't retry — they go back to writing it themselves.

The best coding assistant is the one you cannot use

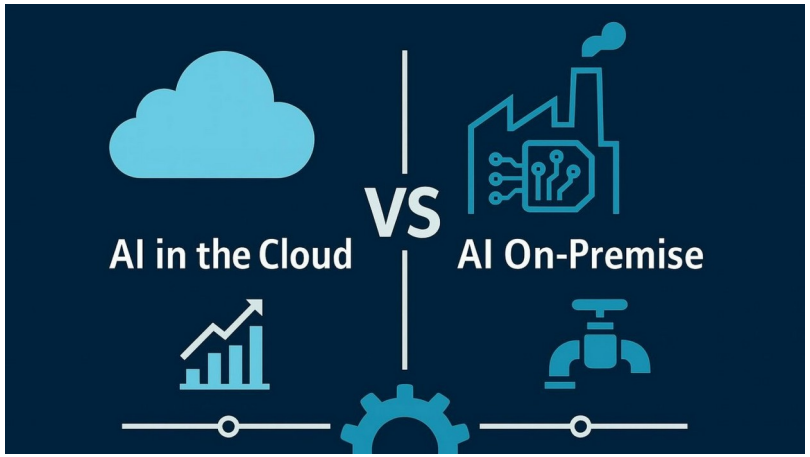
Claude, ChatGPT / Codex, Gemini — genuinely excellent at code. But they are cloud services: every prompt and every code snippet leaves your network.

HPC codebases are proprietary, often export-controlled.

For many code owners, confidentiality outweighs any productivity gain — so the most capable tools are simply off-limits.

100%

of today's frontier coding assistants are cloud-hosted — there is no on-prem Claude or GPT



RELATED WORK

What the community has tried – and the gap

Benchmark

ParEval — Nichols et al. (HPDC 2024)

420 parallel-coding tasks across OpenMP, MPI, CUDA, HIP, Kokkos. LLMs degrade sharply on parallel code.

Pipeline

LASSI — Dearing et al. (2024)

Self-correcting OpenMP ↔ CUDA translator. ~80% executable code, but with multi-iteration overhead.

Fine-tuning

HPC-Coder-V2 — Chaturvedi et al. (2025)

Targeted SFT on parallel code. Confirms data quality > quantity; base models > instruct for HPC.

Translation

Fortran2CPP — Chen et al. (2024)

Multi-turn dialogue dataset, dual-agent generation, 3.3× CodeBLEU improvement.

Our gap: no broad, single-shot, Fortran-first evaluation across both general code and directive translation.

METHODOLOGY

A two-phase benchmark

PHASE 1

164 Fortran coding tasks

FortranHumanEval
(translated HumanEval, function + docstring + tests)

Tests pure Fortran proficiency:
syntax, I/O, allocatables, character handling.

PHASE 2

13 OpenACC → OpenMP target

Translation tasks
vector add, scale, saxpy, reductions, in-place transforms…

Practical motivation:
in-house OpenACC codes need OpenMP-target portability.

A two-phase benchmark, single-shot

PHASE 1

Prompt exemple :

Write a Fortran 90 program that returns the decimal part of a given positive floating-point number.

```
// signature
float truncate_number(float number)

// example  →  expected
3.5         →  0.5
1.33        →  0.33
123.456     →  0.456
```

PHASE 2

Prompt exemple :

Translate this OpenACC kernel to OpenMP target offload (Fortran 90):

```
s = 0.0
!$acc parallel loop reduction(+:s)
do i = 1, n
  s = s + abs(a(i))
end do
```

Output a complete program that reads n and array a, runs on the GPU, and prints the L1 norm.

EVALUATION PIPELINE

Containerised, reproducible, hands-off

HARDWARE

NVIDIA DGX Spark

Grace Blackwell · 128 GB
Compact on-prem inference box.
Designed for multi-model serving
(not exercised here).

120 B = practical ceiling.
GPU ~95% → energy ≈ time.

Prompt	Inference	Extract	Compile	Run+Check
task template + signature	Ollama / API T = 0.7	strip fences, find subroutine	gfortran / nvfortran -mp=gpu	test inputs, output diff

Each task scored on three binary indicators:

$c_i \in \{0,1\}$	$r_i \in \{0,1\}$	$o_i \in \{0,1\}$
compiles	runs	output correct
weight 0.5	*weight 0.3*	*weight 0.2*

All experiments run in two Docker containers (Ollama + evaluator) — fully reproducible.

SCORING

Difficulty-weighted, with a brittle-model penalty

Pass-rate alone is misleading: two models can both clear 30% yet behave very differently. Easy tasks must count.

PER-TASK SCORE

$$s_i = \max(1, 10 \times (0.5\, c_i + 0.3\, r_i + 0.2 o_i))$$

Three binary indicators per task: compiles, runs, output correct. Compilation weighted highest.

DIFFICULTY WEIGHT

$$w_i = 1 + (D_i - 1) / 9$$

Manual difficulty $D_i \in \{1..10\}$ per task. Hard tasks weighted up to 2×.

EASY-TASK PENALTY

$$P = \sum (1 - d_i) \cdot ((10 - s_i) / 10)^2$$

Quadratic penalty for failing easy tasks. A model that misses simple cases is unfit for production.

GLOBAL SCORE

$$S_{global} = \bar{S} - 0.15 \cdot P$$

Difficulty-weighted mean minus the easy-task penalty. Higher is better.

WHAT WE TESTED

45 models, 4 families, 1.5 B → 500+ B parameters

35	10	1.5B – 500+B
open-weight (local)	commercial (OpenRouter)	parameter range (incl. MoE)

Fortran-specialized	*FortranCoder-DS-6.7B, FortranCodeGen-3B*
HPC-focused	*HPC-Coder-V2-6.7B*
Code-specialized	*Qwen2.5-Coder-32B, DeepSeek-Coder-33B, StarCoder2, CodeLlama, …*
General instruct	*Llama 3.3 70B, GPT-OSS 120B, Qwen2.5-32B, Mixtral 8×22B, …*
Commercial frontier	*Claude Opus 4.5, Sonnet 4.5, GPT-5.2, Gemini 3.1 Pro, Grok 4.1, …*

Selection: drawn from the litterature, capped by GPU capacity. *120 B is the practical ceiling on a DGX Spark — we tried larger, it doesn't fit.*

DISCLAIMER

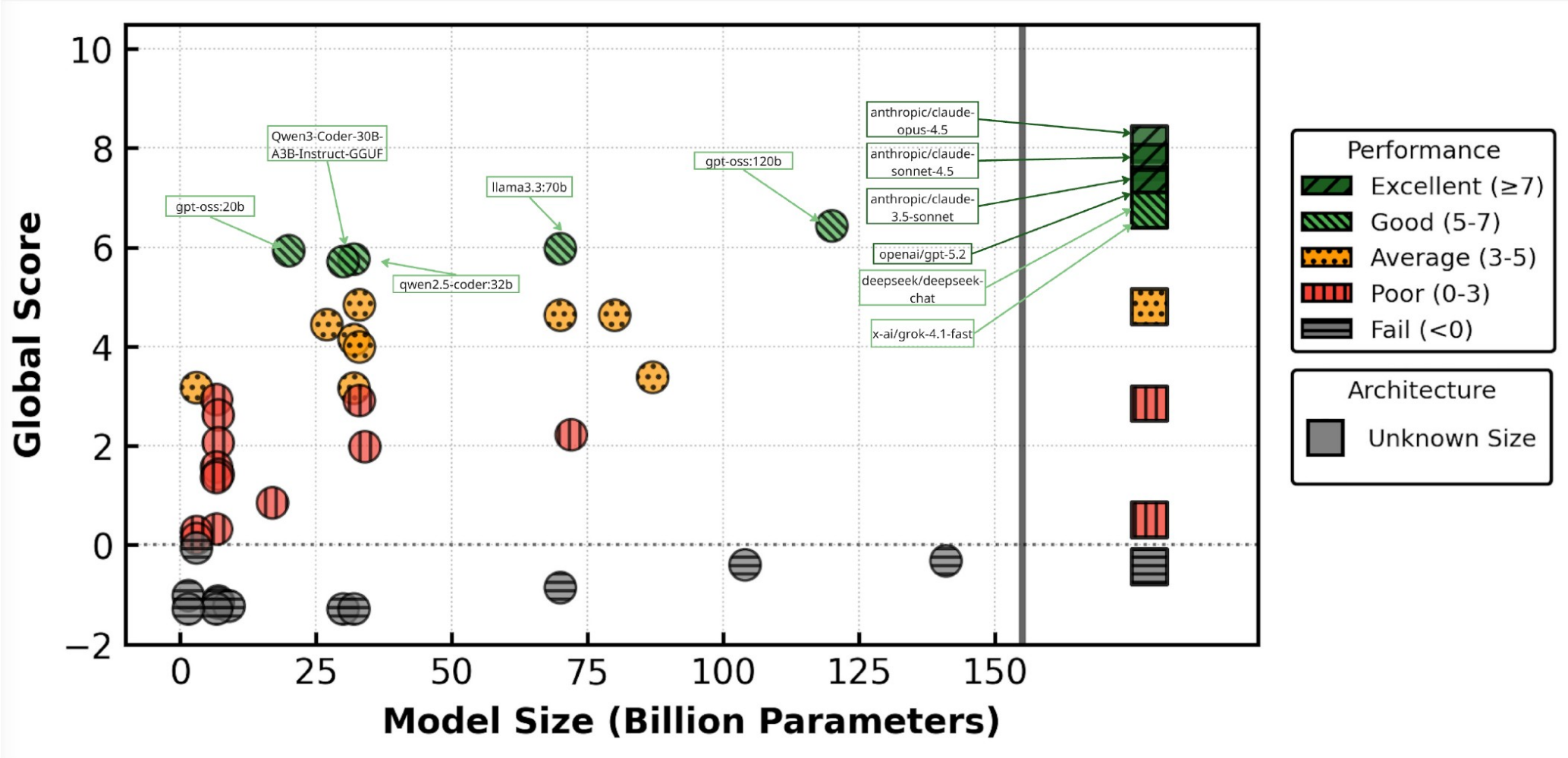
The models will age. The framework won't.

Between paper submission and this talk, several of the models we tested have already been superseded. Treat the rankings the way you'd treat a TOP500 snapshot — accurate at the date, expected to move.

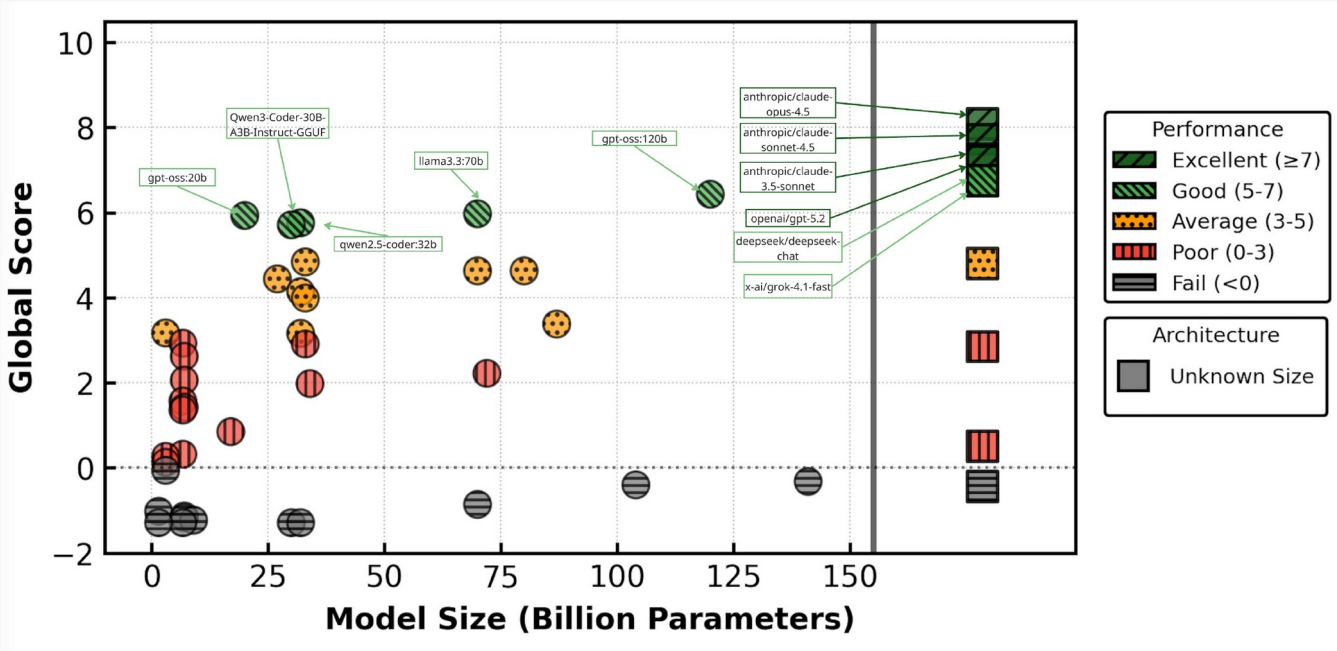
WHAT AGES QUICKLY		WHAT KEEPS WORKING	
Specific rankings	Claude Opus 4.5, GPT-5.2, GPT-OSS 120B…	**Methodology**	Single-shot, difficulty-weighted, easy-task-penalised scoring.
Exact pass rates	60.4% · 41.5% · 13/13 — by next year, all different.	**Pipeline**	Two Docker containers — Ollama + evaluator. Re-runnable on any new model.
Cost figures	$0.001 per correct answer, < $15 to benchmark — drifting fast.	**Benchmark suite**	164 Fortran tasks + 13 OpenACC → OpenMP directive translations.
Hardware ceiling	~120 B parameters on a DGX Spark — a one-year envelope.	**Constraint**	Privacy and IP. The on-prem requirement isn't going away.

⚠ *Re-run the framework whenever a new model lands !*

Size doesn't predict skill



Size doesn't predict skill



1 **The gap is real**

Best open-weight ≈ 41.5% pass (GPT-OSS 120B/DeepSeek-Chat). Best commercial = Claude Opus 4.5 at 60.4%.

2 **Mid-sized local models punch up**

Qwen2.5-Coder-32B and Qwen3-Coder-30B clear the “good” line — encouraging for on-prem deployment.

3 **Sometimes closed weights ≠ quality**

Nemotron Ultra 253B and Gemini 3.1 Pro sit near the bottom; raw size or secrecy guarantees nothing.

COMMERCIAL VS OPEN-WEIGHT

Cloud still wins on raw Fortran — mind the gap, and the cost

#	Model	Type	Pass %	S_global
1	**Claude Opus 4.5**	API	**60.4%**	7.43
2	**Claude Sonnet 4.5**	API	**55.5%**	6.95
3	**GPT-5.2**	API	**53.7%**	6.07
4	**DeepSeek-Chat**	local	**41.5%**	5.38
4	**GPT-OSS 120B**	local	**41.5%**	4.86
6	**Llama 3.3 70B**	local	**31.7%**	4.19

Top models on 164 Fortran tasks (Phase 1). "local" = open-weight, runs on-prem.

THE COST REALITY

< $15 to benchmark all 10 commercial models on 177 tasks (OpenRouter)

$0.001 per correct answer — DeepSeek-Chat

$0.012 per correct answer — Claude Sonnet 4.5

$0 marginal cost and zero exfiltration — local inference

Today, single-shot Fortran generation still favours cloud — **but the best open-weight is already ~⅔ of the way, improving monthly, and free of exfiltration.**

OpenRouter

The counter-intuitive finding

Mean Fortran pass rate by model family — the more specialized for code, the worse on Fortran.

General-purpose · n = 15	Code-specialized · n = 18	Fortran-specialized · n = 2
16.4%	**11.2%**	**6.4%**
mean pass rate	mean pass rate	mean pass rate
compile rate ≈ 53.4%	*compile rate ≈ 47.2%*	*compile rate ≈ 51.8%*
Llama 3.3 70B, GPT-OSS 120B, DeepSeek-Chat …	Qwen2.5-Coder, DeepSeek-Coder, StarCoder2 …	FortranCoder-DS, FortranCodeGen

Why? Code-specialized models are fine-tuned on Python-heavy corpora — the very habits (dynamic typing, missing IMPLICIT NONE, 0-based indexing) that **break** when generating Fortran.

Hallucination caveat: *Other Fine-tuned candidates hallucinated too badly to evaluate at all — they don't appear in our 45.*

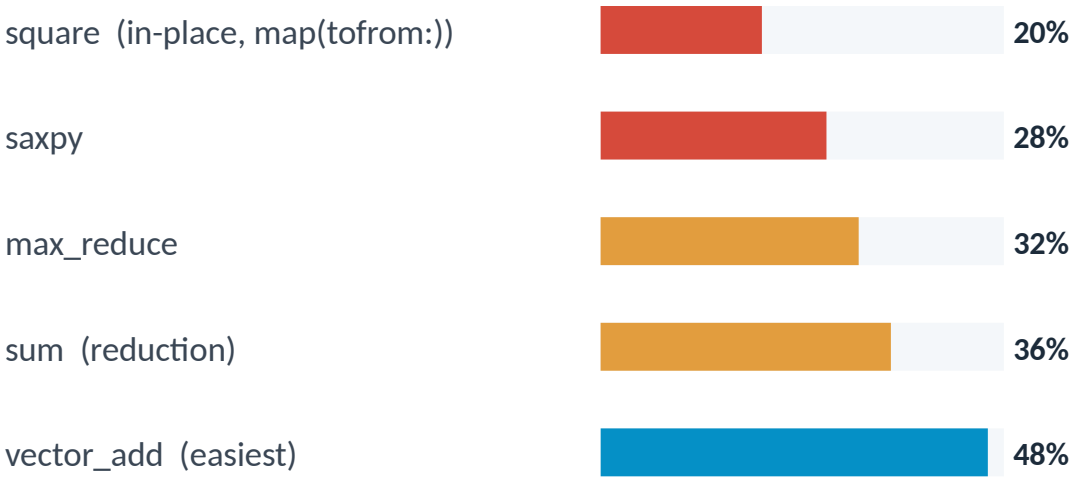
PHASE 2 · OPENACC → OPENMP TARGET

Directive translation is already tractable — locally

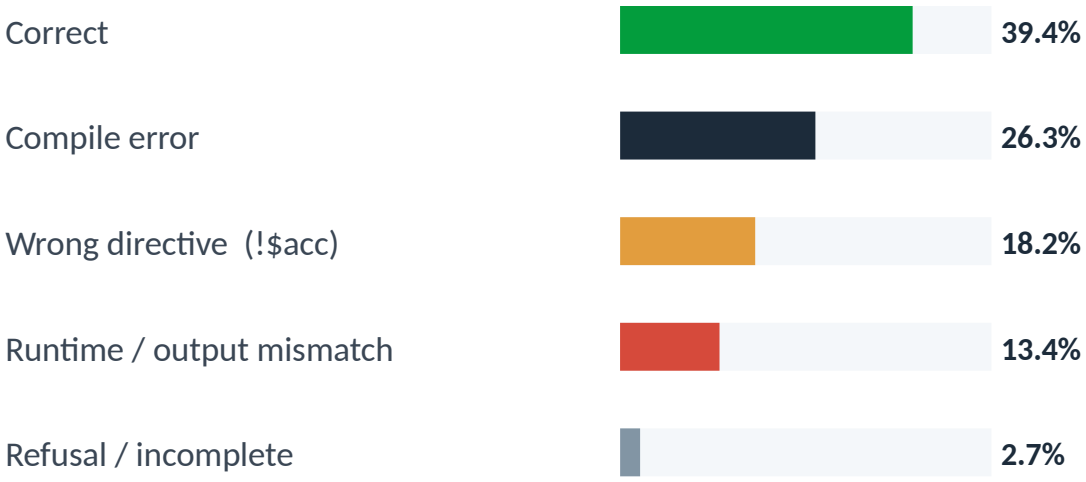
13 / 13 perfect score for **GPT-OSS 120B** — open-weight, running on-prem.

7 models reach ≥ 90% · commercial mean 82.3% vs open-weight mean 28.7%

HARDEST TASKS · cross-model pass rate



FAILURE MODES · 650 attempts, all models



Reductions and in-place transforms remain the wall — even strong models miss the map-clause direction. Wrong-directive failures (emitting !$acc instead of !$omp) appear exclusively on open-weight models.

ROADMAP

From evaluation to an invariant-first workflow

1	**Apply & benchmark on real HPC software**	Move beyond benchmark exemples to representative in-house codes, with performance-inclusive metrics.	
2	**Fortran fine-tuning from our own codebase**	Curate datasets from scientific libraries and proprietary code — the data the public models never saw.	NEXT PAPER
3	**Dedicated agents + workflow automation**	Custom and open “skills”, deterministic prompt pipelines, served results.	AFTER
4	**GPU-targeted**	Build GPU-oriented setups like OpenACC to OpenMP translation and so on…	

Direction of travel: from legacy Fortran towards performance-portable code — with AI in the loop, on-prem.

THE MESSAGE

For closed-source code, local LLMs deserve a serious look

1

When confidentiality is non-negotiable, local isn't second-best — it's the only option.

And it is viable today: a desk-side box, no exfiltration, no per-token bill.

2

The quality gap to frontier cloud is real but narrowing fast.

Best open-weight already ~⅔ of Claude on Fortran generation — better on constrained translation.

3

The levers compound on the same hardware.

Right model + fine-tuning + agents & skills + disciplined prompts close much of the remaining gap.

⚠ **Again —** this snapshot reflects models available in early 2026. The field moves faster than the paper review cycle.

Thank you

thibault.marzlin@cerfacs.fr · mohamed.ghenai@cerfacs.fr · antoine.dauptain@cerfacs.fr · cerfacs.fr

Founding industrial & institutional partners