ParEval-Repo: A Benchmark Suite for Evaluating LLMs with Repository-level HPC Translation Tasks

Josh H. Davis, Daniel Nichols, Ishan Khillan, Abhinav Bhatele

PARALLEL SOFTWARE AND SYSTEMS GROUP

INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

UNIVERSITY OF MARYLAND

The Team

Joshua H. Davis

Daniel Nichols

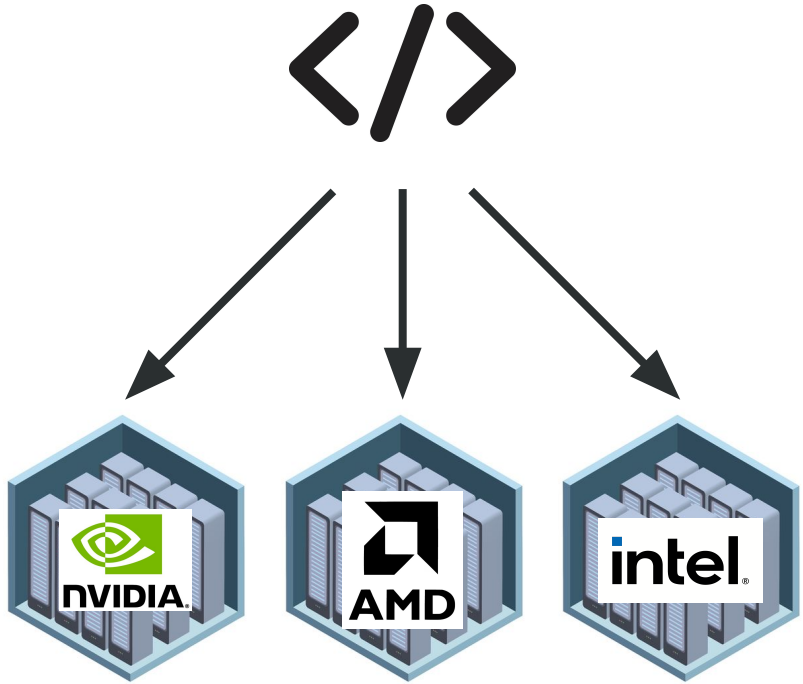
Ishan Khillan

Abhinav Bhatele

Why Automate Translation for HPC?

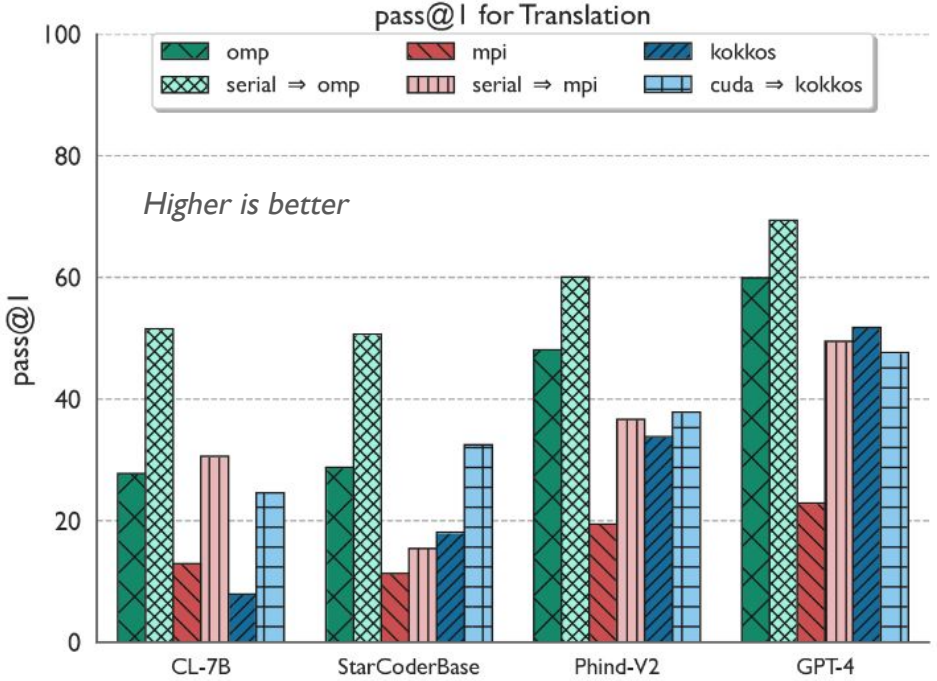
- Three different GPU vendors in top ten supercomputers
- Maximizing access requires using portable programming models
- But, manually porting is time-consuming and tedious!

Research Question: Can LLMs help to translate HPC code repositories into portable GPU programming models?



Background for LLM Parallel Code Translation

- ParEval found that parallel code tasks are uniquely difficult
- Function-level LLM translation of parallel code is more feasible than generation
- But we don't know if this will scale to full repositories



D. Nichols, J. H Davis, et al. ***"Can Large Language Models Write Parallel Code?"****, HPDC 2024*

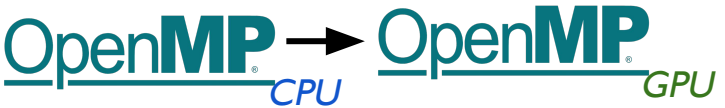
Introducing ParEval-Repo: Translation Tasks

Across codes chosen, using three programming model pairs:

CUDA to OpenMP Offload
- Non-portable to directive-based portable

CUDA to Kokkos
- Non-portable to C++ abstraction-based portable

OpenMP Threads to OpenMP Offload
- CPU portable to GPU portable



Introducing ParEval-Repo: Small Cases

Starting small: three minimized codes to test capability to handle simple repos

All three include a Makefile

1. **nanoXOR**: 1 source file
2. **microXORh**: 1 source file, 1 header file
3. **microXOR**: 2 source files, 1 header file

XOR kernel:

```
__global__ void cellsXOR(const int *input,
                         int *output, size_t N) {
  int i = blockIdx.y * blockDim.y + threadIdx.y;
  int j = blockIdx.x * blockDim.x + threadIdx.x;
  if (i < N && j < N) {
    int count = 0;
    if (i > 0   && input[(i-1)*N + j] == 1) count++;
    if (i < N-1 && input[(i+1)*N + j] == 1) count++;
    if (j > 0   && input[i*N + (j-1)] == 1) count++;
    if (j < N-1 && input[i*N + (j+1)] == 1) count++;
    output[i*N + j] = (count == 1) ? 1 : 0;
  }
}
```

Introducing ParEval-Repo: Larger Cases

Three larger cases are taken from public codes – want to avoid codes that already have ports to minimize data contamination

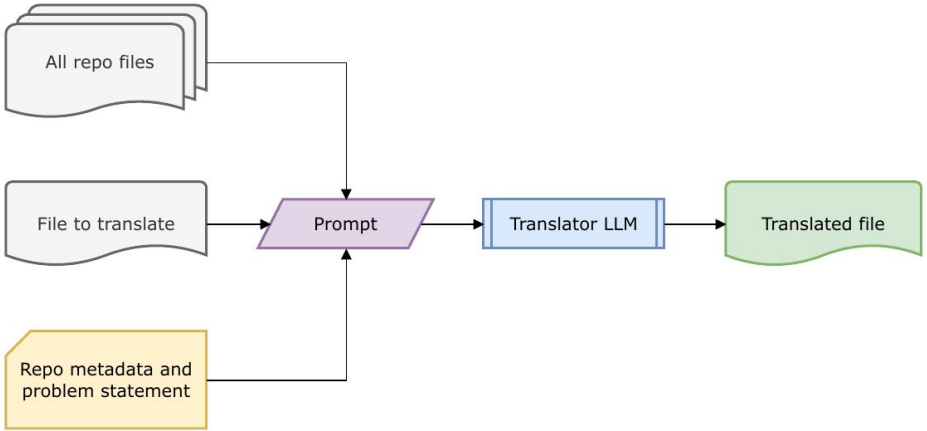
4. **SimpleMOC-kernel**: SimpleMOC proxy, neutron flux attenuation. Has external dependency on cuRAND.
5. **XSBench**: OpenMC proxy, macroscopic cross-section lookup. Has public ports to OpenMP and Kokkos – will show impact of possible data contamination
6. **llm.c**: CUDA implementation of LLM pretraining. We reduced the repo complexity slightly.

ParEval-Repo Translation Benchmark Suite

App. name	# of source lines	Cyclomatic complexity	# of files	OpenMP Offload	Kokkos
nanoXOR	109	33	2	X	X
microXORh	127	33	3	X	X
microXOR	133	33	4	X	X
SimpleMOC-kernel	780	59	6	X	X
XSBench	2449	264	9	✓	✓
llm.c	3039	360	7	X	X

Translation Techniques: Naive

Naive translation technique: translate file-by-file in arbitrary order



Problem: exceeds LLM input context window for all but the smallest apps

Sample prompt:

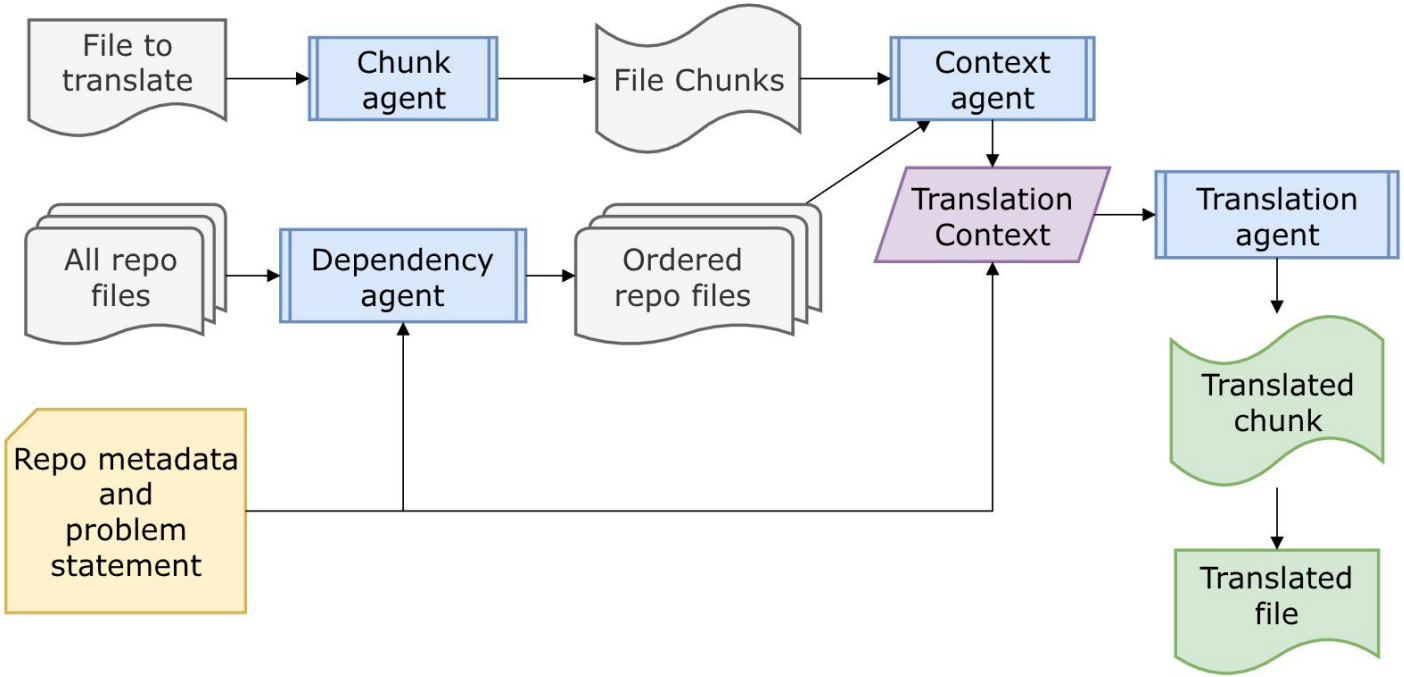
```
You are a helpful coding assistant. You are helping a
software developer translate a codebase from the CUDA
execution model to the OpenMP Offload execution model...

Below is a codebase written in the CUDA execution model...
Here is the file tree of the entire repository:

<file tree>

Here is the code for each file in the codebase:

Makefile
...

src/main.cpp
...

Translate the src/main.cpp file to the OpenMP offload
execution model...
```

Translation Techniques: Agentic

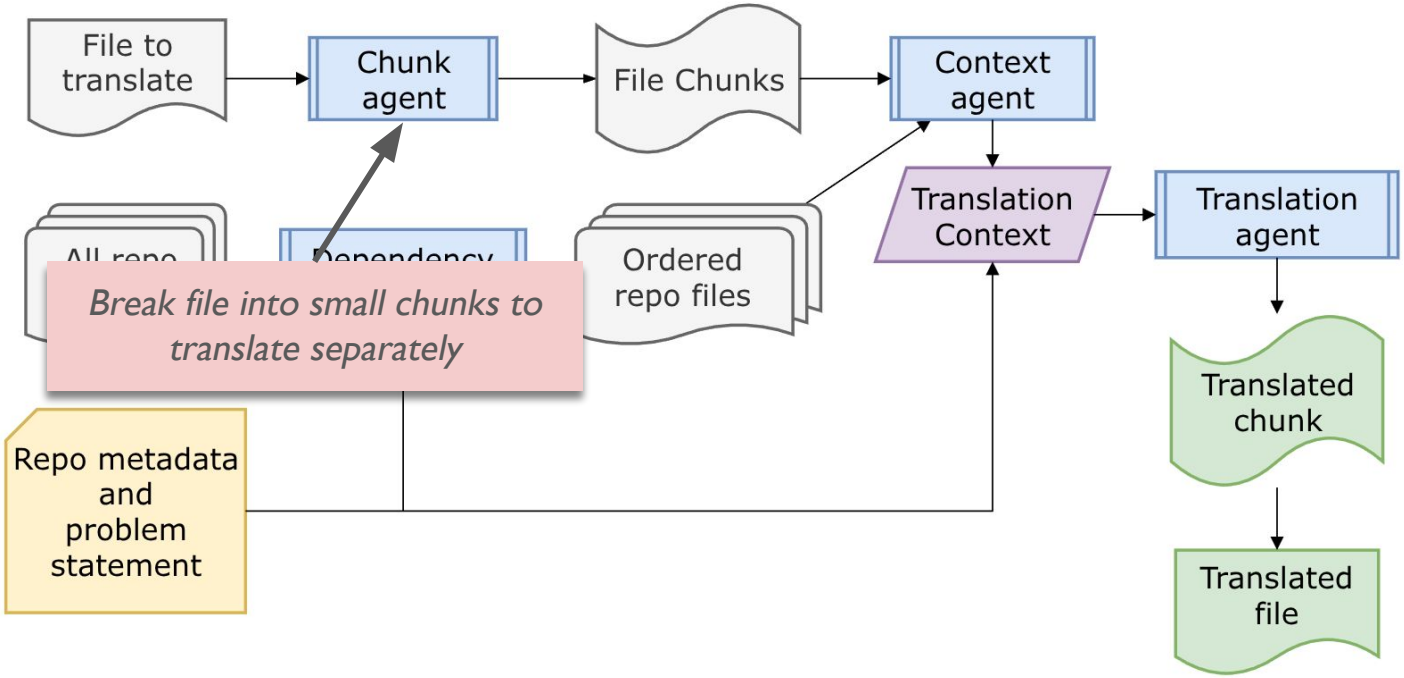
Need to break up the task into pieces that can fit into LLM context window

- Break each file into separate "chunks" that fit in context
- Order translation of files by dependency structure
  - Headers before source, source before build files)
- Use a context agent LLM to retrieve context from already-translated files for next translation

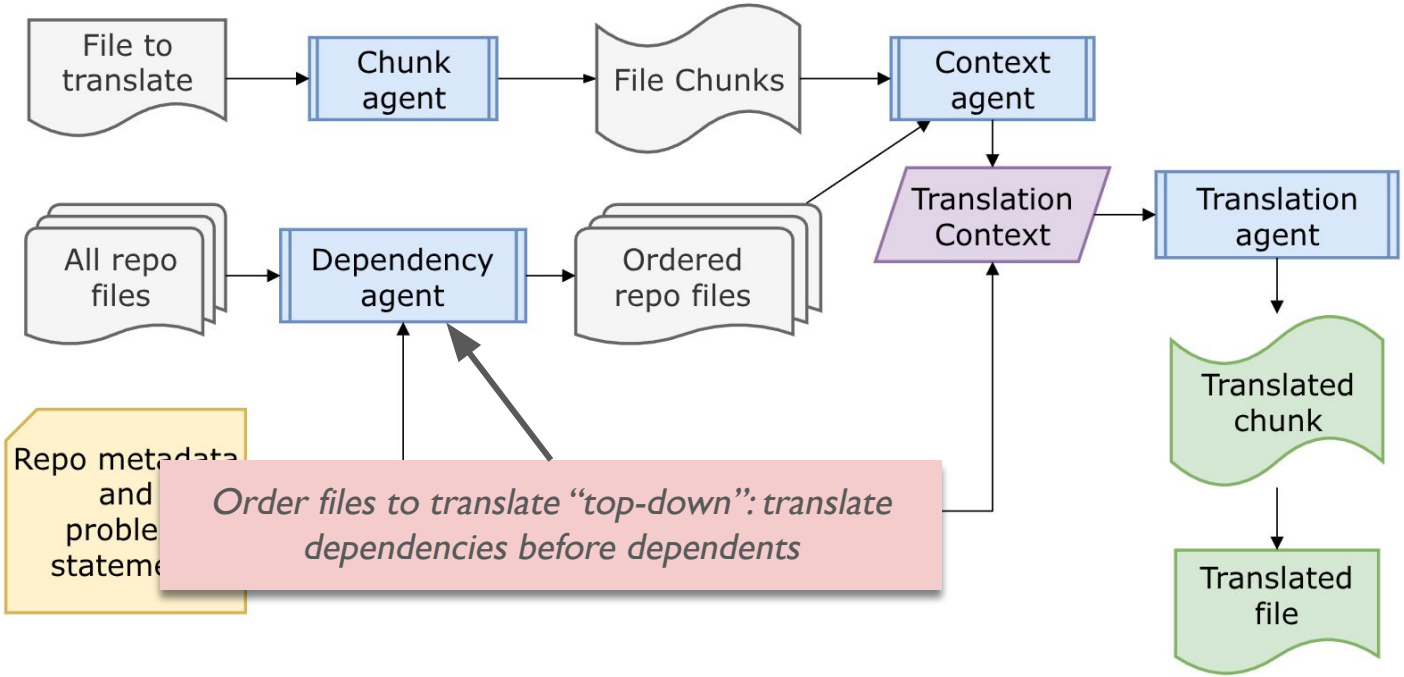
Translation Techniques: Agentic



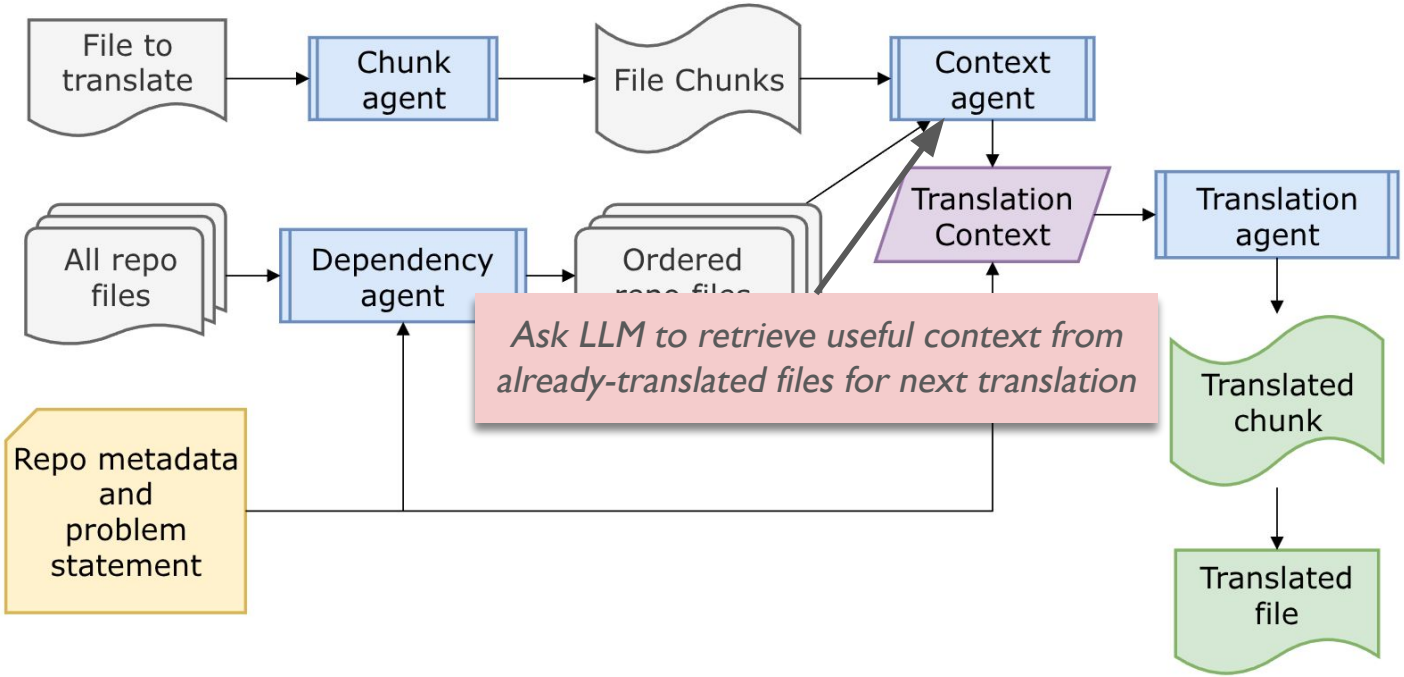
Translation Techniques: Agentic



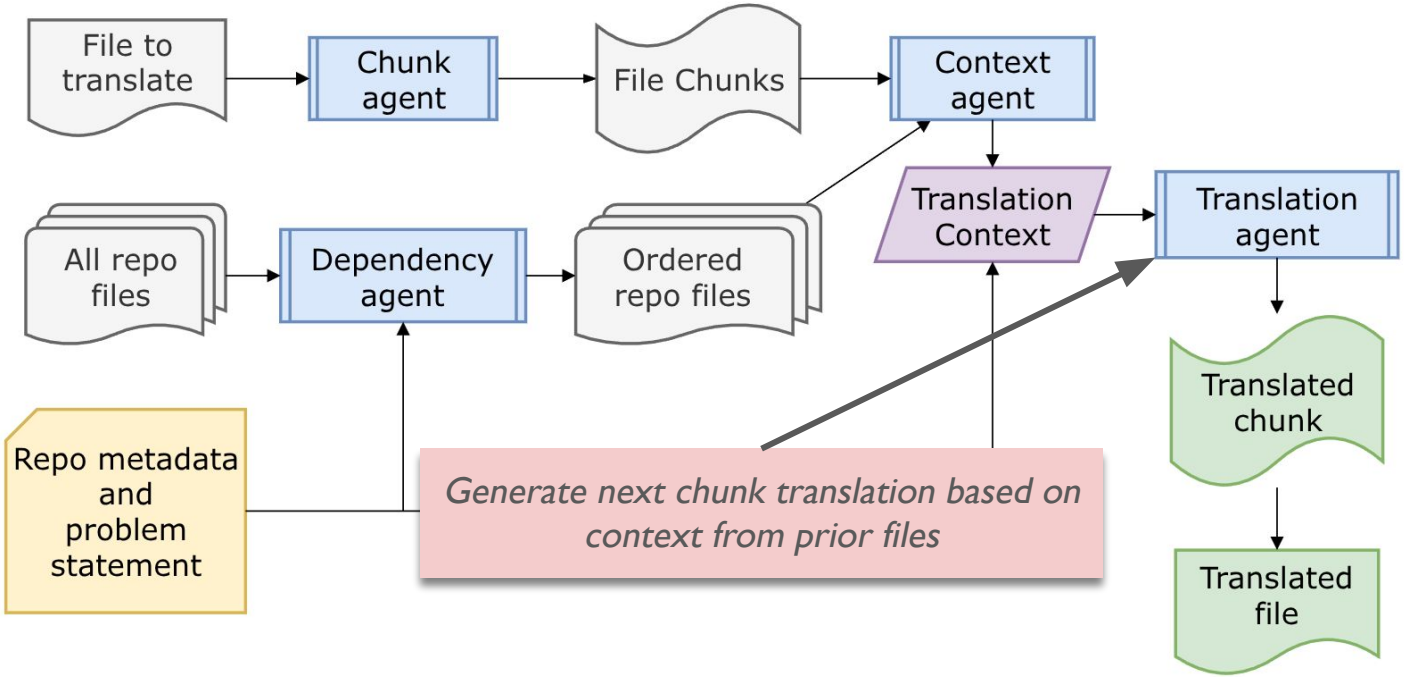
Translation Techniques: Agentic



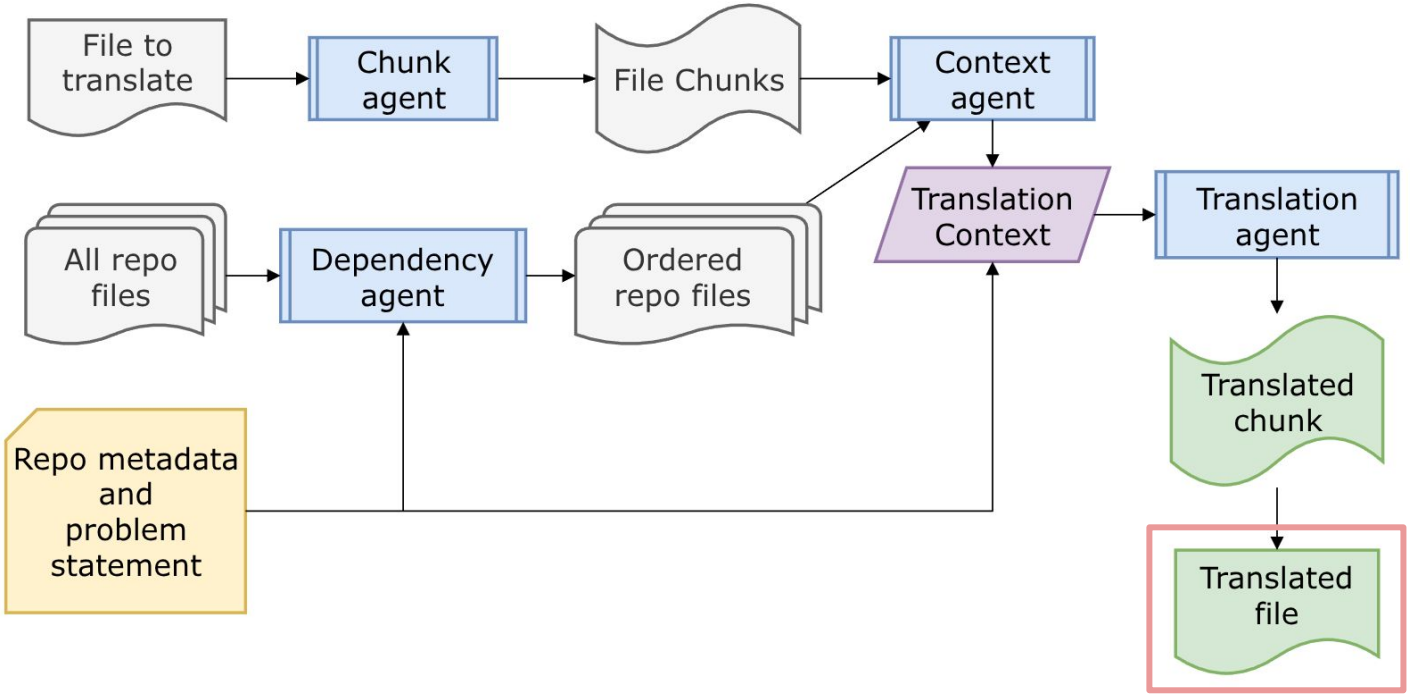
Translation Techniques: Agentic



Translation Techniques: Agentic



Translation Techniques: Agentic



Translation Techniques: SWE-agent

- SWE-agent is a state-of-the-art software engineering (SWE) LLM framework
- Can call tools, write tests, run code, etc.
- Needed to format our translation tasks as Github issues
- Designed for Python: editing tool cannot handle Makefiles due to tabs

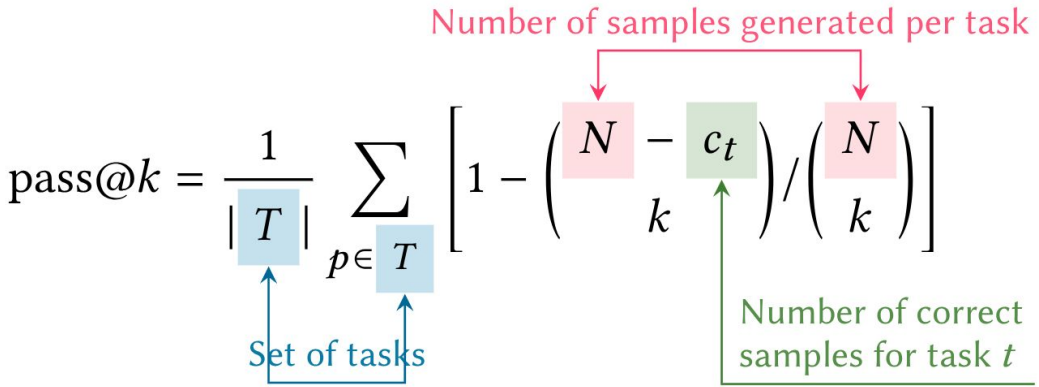
Experimental Setup: LLMs Used

LLM Name	Provider	# of parameters	Open-source?	Reasoning?
Gemini 1.5 Flash	Google	?	Closed	Generic
GPT 4o-mini	OpenAI	?	Closed	Generic
o4-mini	OpenAI	?	Closed	**Reasoning**
Llama 3.3 70B Instruct	Meta	70 billion	**Open**	Generic
QwQ-32B	Alibaba Cloud	32 billion	**Open**	**Reasoning**

Experimental Setup: Evaluation Metrics

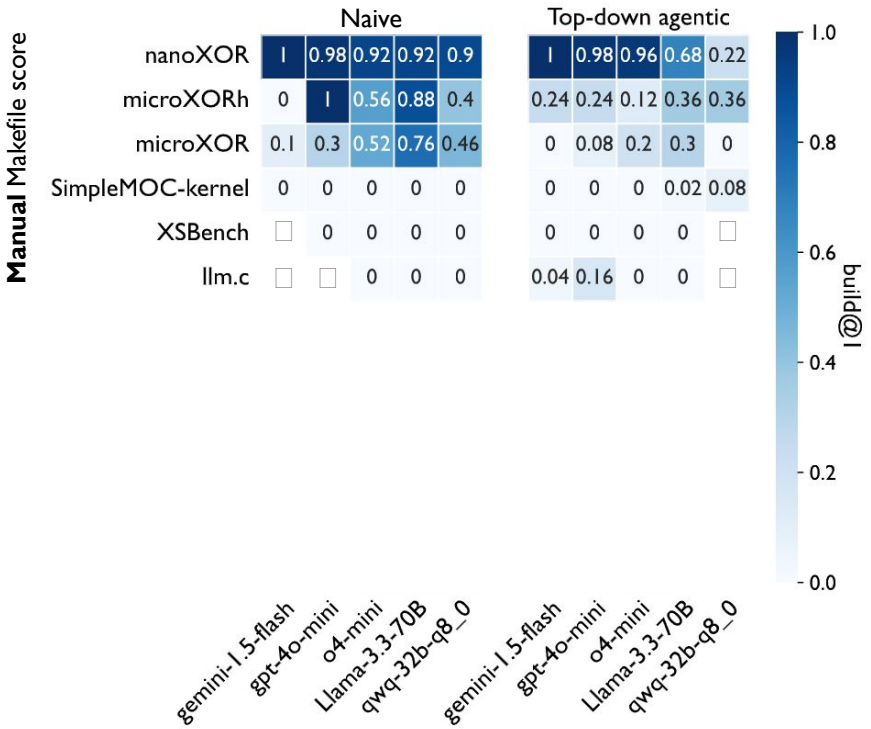
Evaluation Metrics:

- **pass@1** scores: chance of generating correct translation with one attempt
  - Assessed using app's correctness test cases
- **build@1** scores: chance of generating compilable translation with one attempt



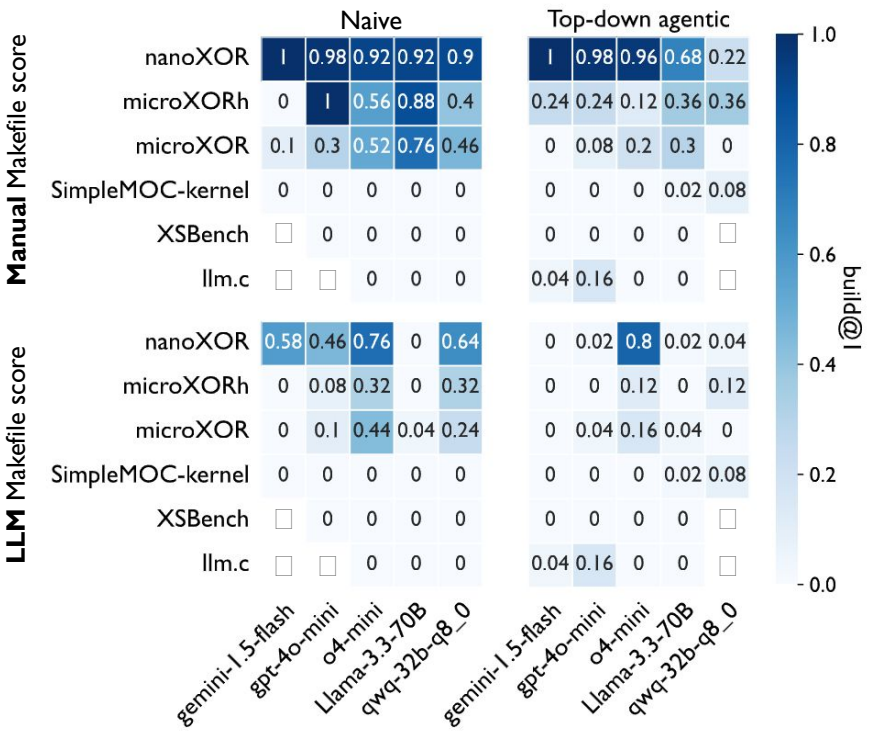
k = 1
N = 20

Build@1 for CUDA to OpenMP Offload Translation



- Minimal success with larger codes
- When the LLM has to translate the build system as well, scores are even lower

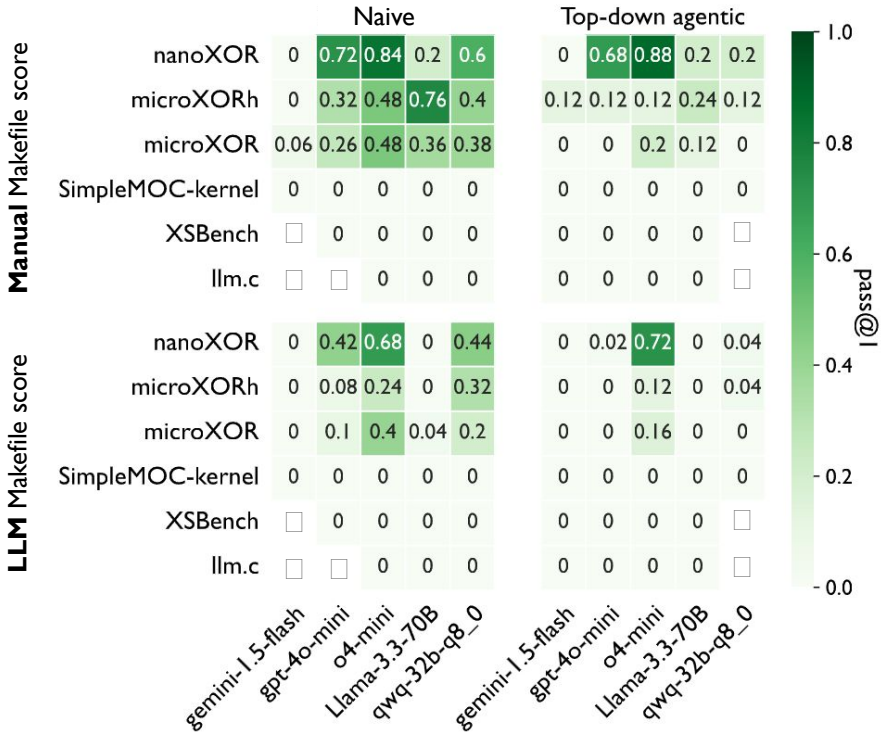
Build@1 for CUDA to OpenMP Offload Translation



NVIDIA CUDA → OpenMP

- Minimal success with larger codes
- When the LLM has to translate the build system as well, scores are even lower

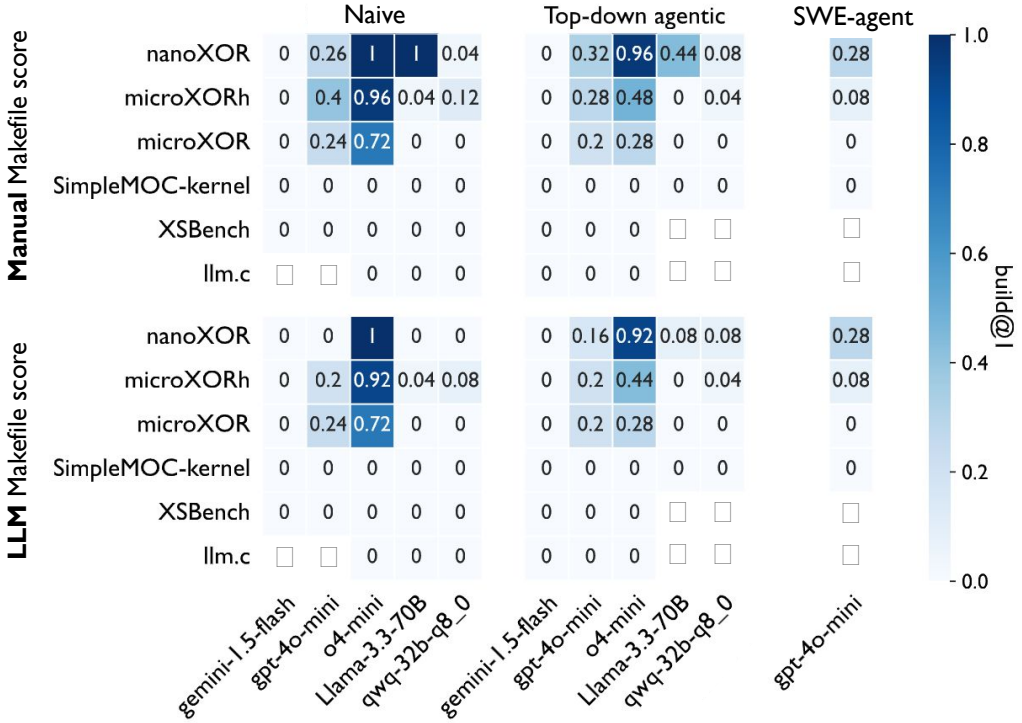
Pass@1 for CUDA to OpenMP Offload Translation



NVIDIA CUDA → OpenMP

- Significant correctness cliff beyond 1-3 files
- **o4-mini naive** achieves overall best pass@1 scores
- Naive often beats agentic

Build@1 Results for CUDA to Kokkos



- Kokkos is even harder
  - Requires generating CMakeLists.txt
- Even with CMake provided, LLMs struggle to write compilable Kokkos
- SWE-agent underperforms

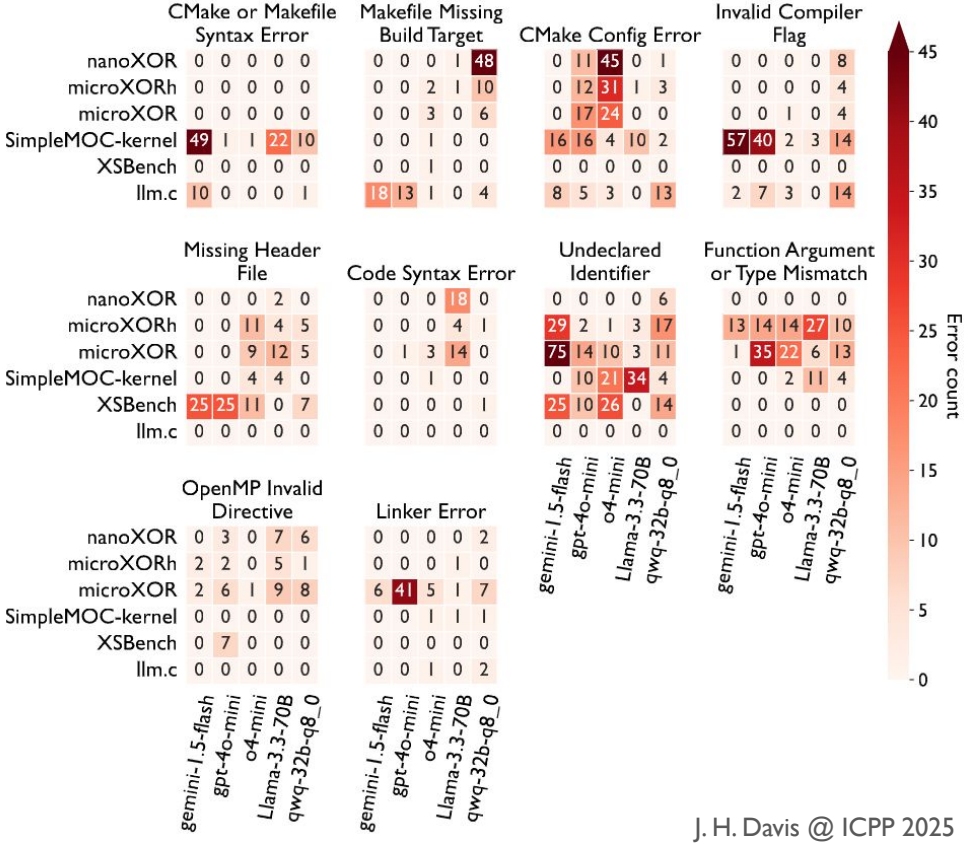
Error Clustering Across Translation Tasks

What are LLMs struggling with in generating buildable translations?

- Conducted clustering analysis of build outputs
- Embedded outputs to vectors with word2vec
- Clustered vectors using DBSCAN with manually tuned hyperparameters
- Manually reviewed clusters to set names, merge and split some clusters as needed

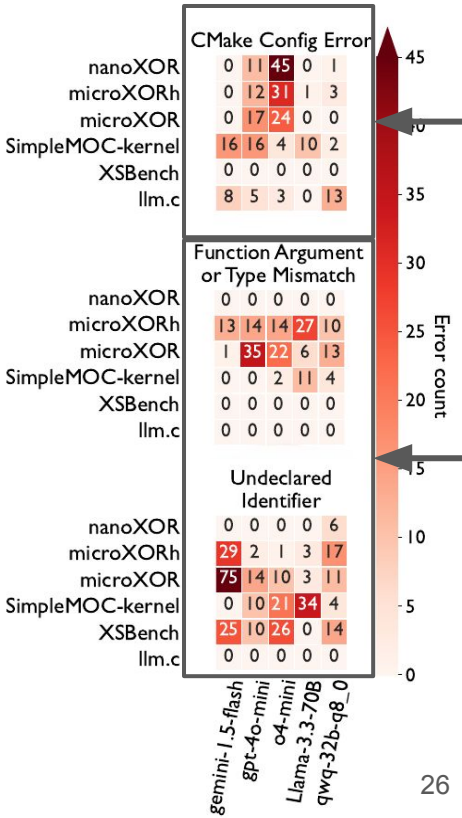
Error Clustering Across Translation Tasks

What are LLMs struggling with in generating buildable translations?



Error Clustering Across Translation Tasks

What are LLMs struggling with in generating buildable translations?



Setting up CMakeLists.txt correctly to build with the Kokkos library

Maintaining consistent interfaces across files and including relevant dependencies

Translation Cost Analysis

Method and LLM	nanoXOR	microXORh	microXOR
Naive o4-mini	$0.03	$0.03	$0.05
Naive Llama-3.3	36 node-mins.	4 node-mins.	5 node-mins.

- **LLM translation is expensive**
- We derive expected tokens used with

$$E_\kappa = \left(\frac{1}{\text{pass@1}} \right) \cdot \kappa$$

Average token cost per generation for the task

Single generation correctness chance

- Convert this to node hours or API cost ($) as relevant for the model

Conclusion and Future Work

- Employing LLMs to automate application translation to new programming models has tremendous potential for developer productivity
- But, state-of-the-art LLMs struggle to correctly translate full applications
- Translating build systems and cross-file dependencies are most significant hurdles
- Opportunities for improvement: building a dataset for fine-tuning or improving prompt context, or building a more sophisticated agent approach

← ParEval-Repo is on GitHub!

Contact: jhdavis@umd.edu

jhdavis@umd.edu

ParEval-Repo Translation Benchmark Suite

App. name	# of source code lines	Cyclomatic complexity	# of files	OpenMP Threads	OpenMP Offload	CUDA	Kokkos
nanoXOR	109	33	2	✓	?	✓	?
microXORh	127	33	3	✓	?	✓	?
microXOR	133	33	4	✓	?	✓	?
SimpleMOC-kernel	780	59	6		?	✓	?
XSBench	2449	264	9	✓	✓?	✓	✓?
llm.c	3039	360	7		?	✓	?

Port already exists

Port doesn't exist, will translate