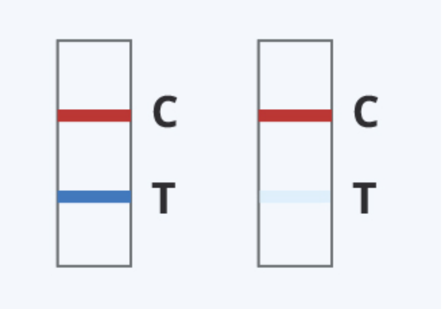
Did you noticed something odd or unnatural in the talks so far?

[1] No AI, LLMs, ChatGPT so far

We will fix this now!

[2] No negativ results



Fine-Tuning LLMs for Predicting Energy Consumption of Configurable Software Systems

Nicole Heinimann, Norbert Siegmund
Chair of Software Systems
Applied AI and Big Data: Software Engineering

UNIVERSITÄT LEIPZIG

ScaDS.AI
DRESDEN LEIPZIG

SPONSORED BY

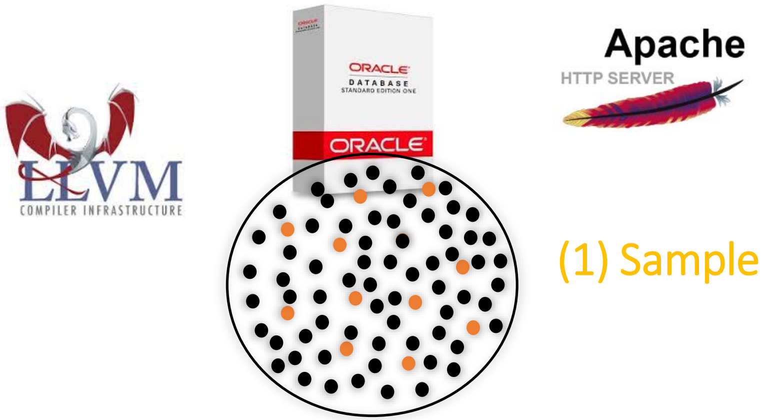
Bundesministerium
für Bildung
und Forschung

FKZ: ScaDS.AI

SACHSEN

Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

Current State



(1) Sample

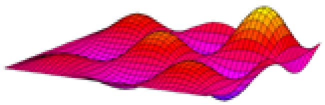
How to scale measurements?

How to obtain reliable measurements?

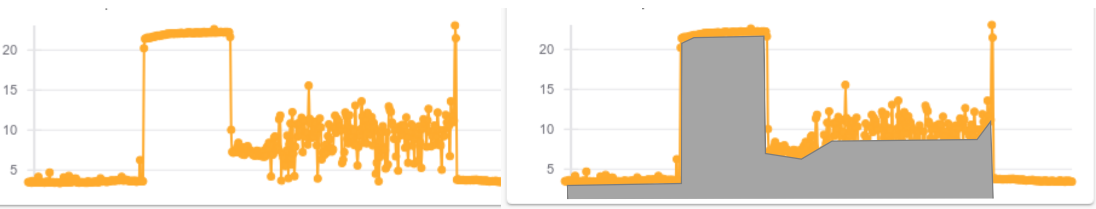
How to obtain fine-grained measurements?

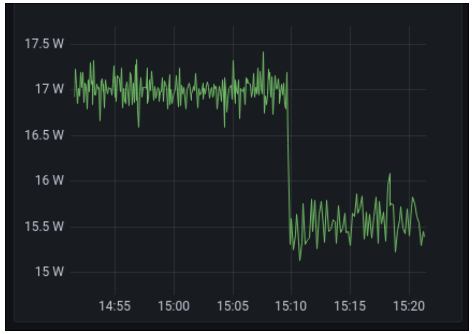
(2) Measure energy

$\Pi : C \rightarrow \mathbb{R}$

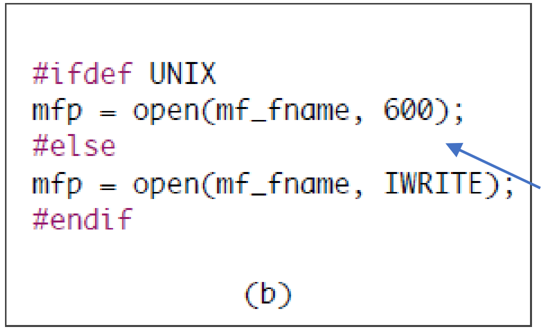


(3) Learn model





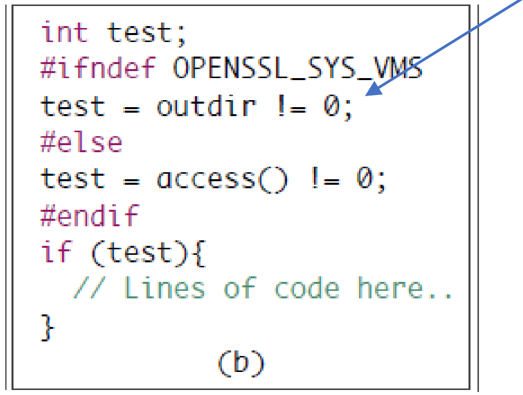
Limits of Granularity



```
#ifdef UNIX
mfp = open(mf_fname, 600);
#else
mfp = open(mf_fname, IWRITE);
#endif
```

(b)

Can we measure energy consumption of these code fragements?

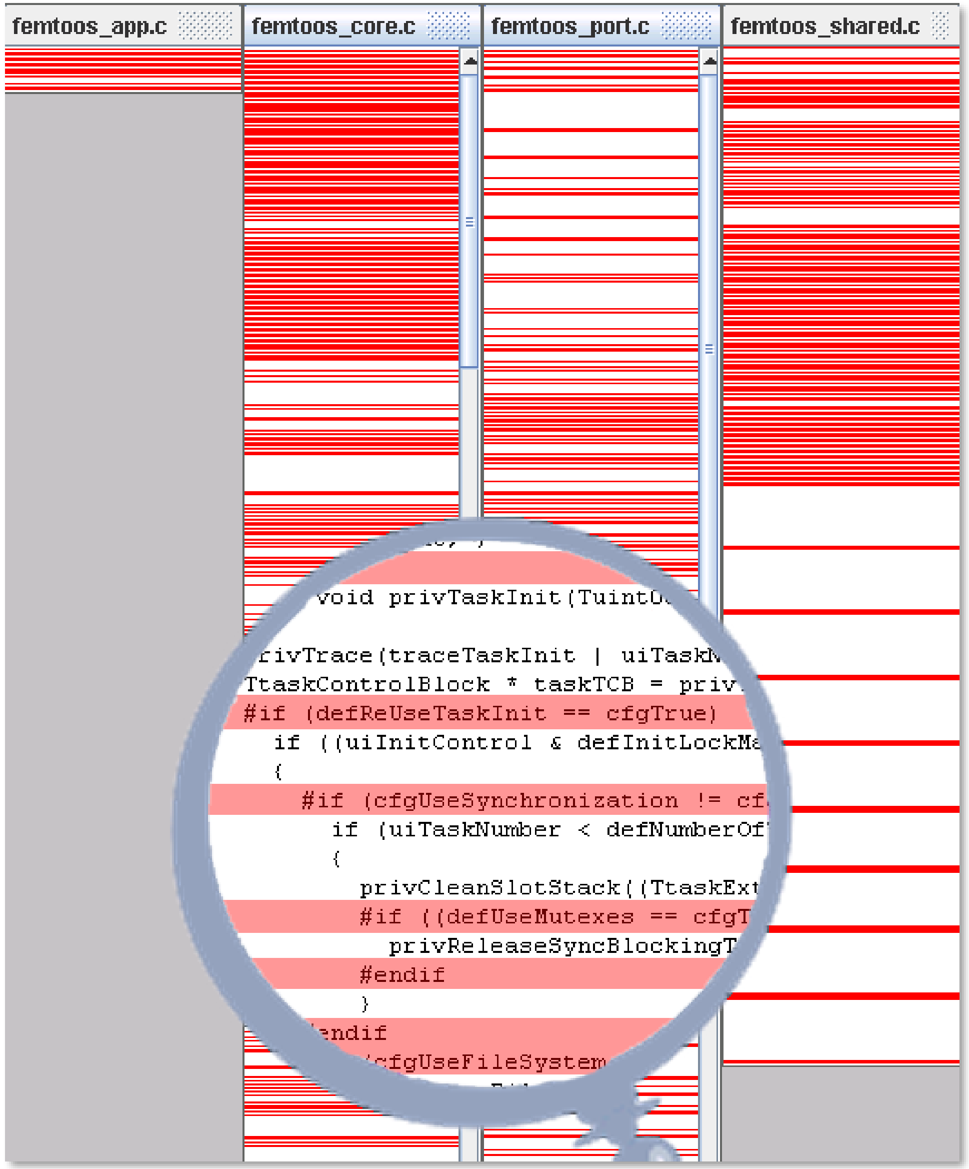


```
int test;
#ifndef OPENSSL_SYS_VMS
test = outdir != 0;
#else
test = access() != 0;
#endif
if (test){
  // Lines of code here..
}
```

(b)

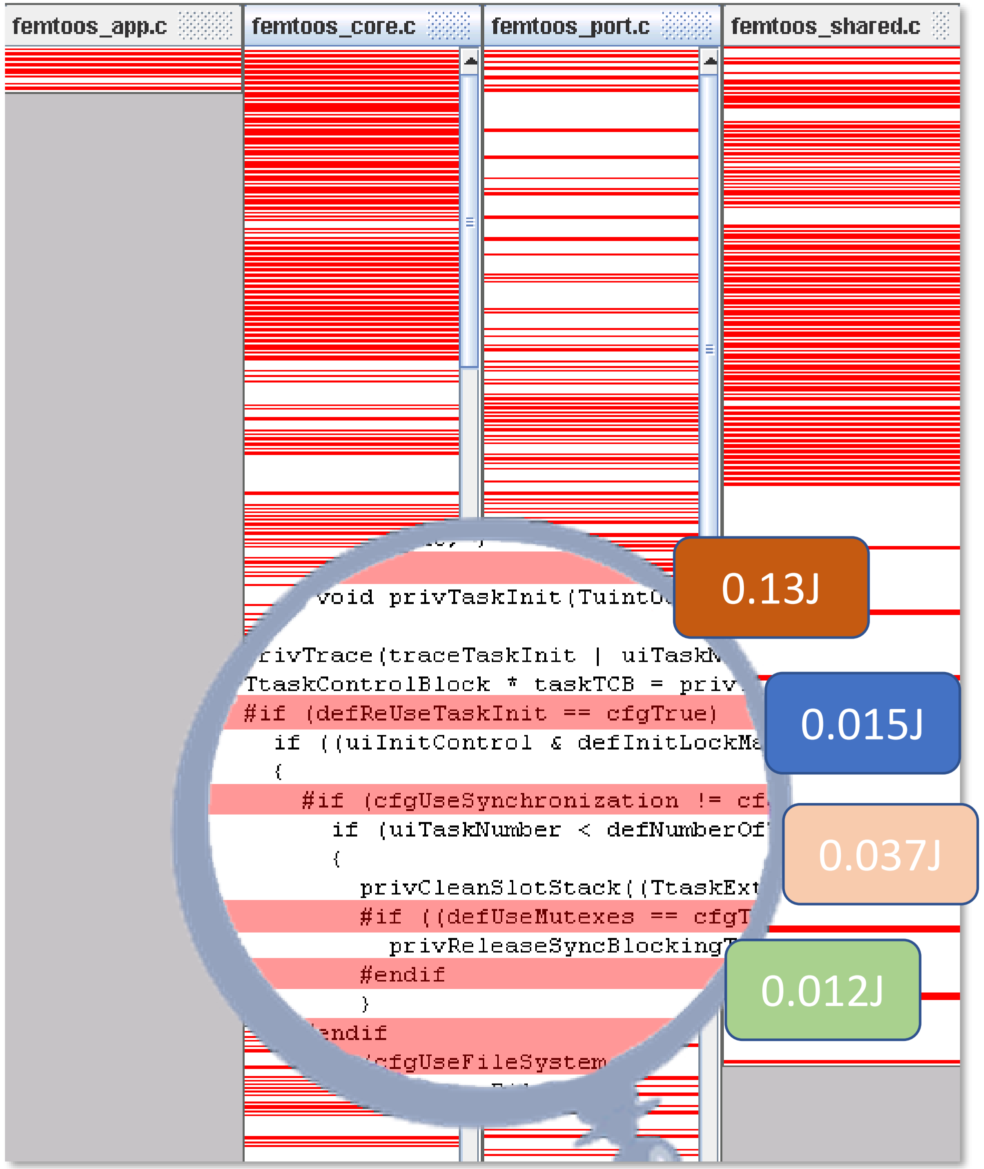
libpng

And now?



```
  void privTaskInit(TuintO
rivTrace(traceTaskInit | uiTaskN
TtaskControlBlock * taskTCB = priv
#if (defReUseTaskInit == cfgTrue)
  if ((uiInitControl & defInitLockMa
  {
    #if (cfgUseSynchronization != cf
      if (uiTaskNumber < defNumberOf
      {
        privCleanSlotStack((TtaskExt
        #if ((defUseMutexes == cfgT
          privReleaseSyncBlockingT
        #endif
        }
  endif
    cfgUseFileSystem
```

The Dream

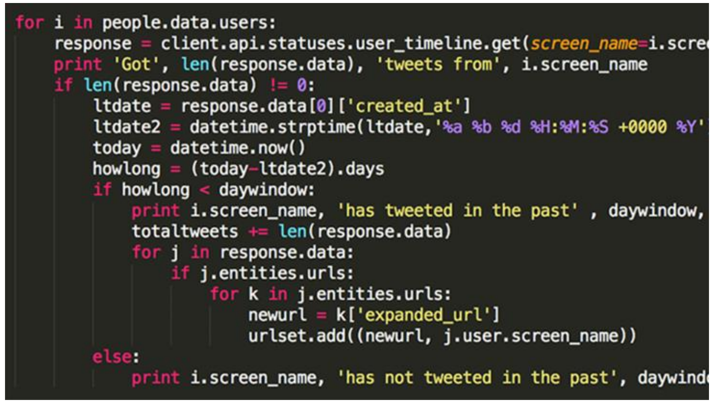


```
   void privTaskInit(Tuint0
 rivTrace(traceTaskInit | uiTaskN
TtaskControlBlock * taskTCB = priv
#if (defReUseTaskInit == cfgTrue)
  if ((uiInitControl & defInitLockMa
  {
    #if (cfgUseSynchronization != cf
      if (uiTaskNumber < defNumberOf
      {
        privCleanSlotStack((TtaskExt
        #if ((defUseMutexes == cfgT
          privReleaseSyncBlockingT
        #endif
        }
  endif
    /cfgUseFileSystem
```

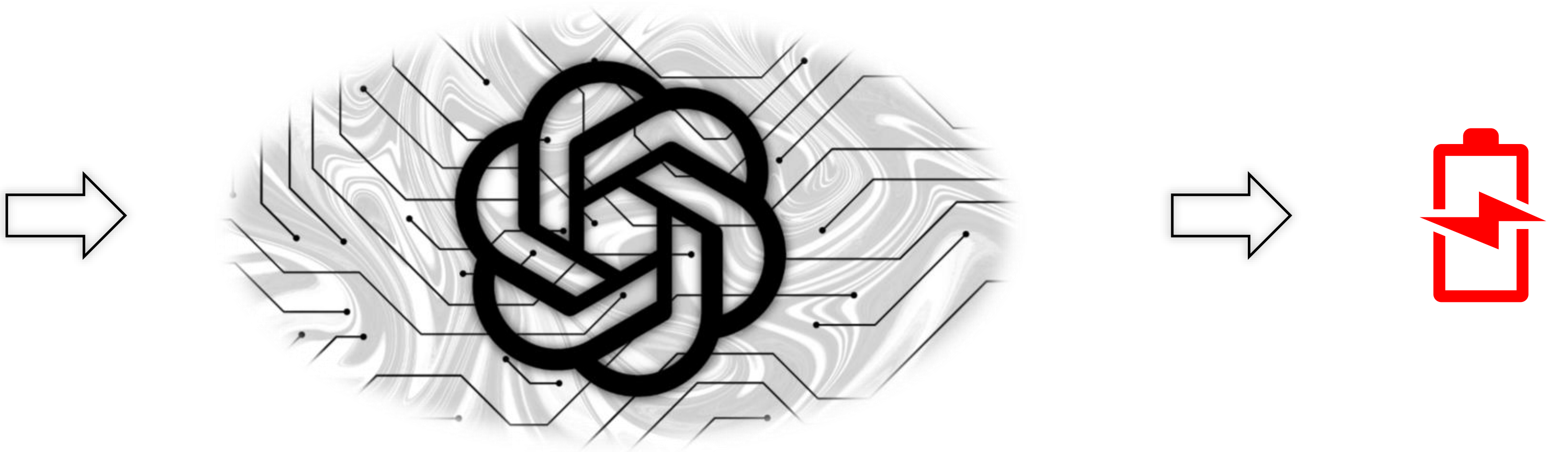
How?   You have read the title… right? ☺

The Idea

LLMs are great for code understanding, generation, and summarization



```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.scre
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate,'%a %b %d %H:%M:%S +0000 %Y'
        today = datetime.now()
        howlong = (today-ltdate2).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past' , daywindow,
            totaltweets += len(response.data)
            for j in response.data:
                if j.entities.urls:
                    for k in j.entities.urls:
                        newurl = k['expanded_url']
                        urlset.add((newurl, j.user.screen_name))
        else:
            print i.screen_name, 'has not tweeted in the past', daywind
```



Can we adapt and fine-tune an LLM to estimate energy consumption of given code?

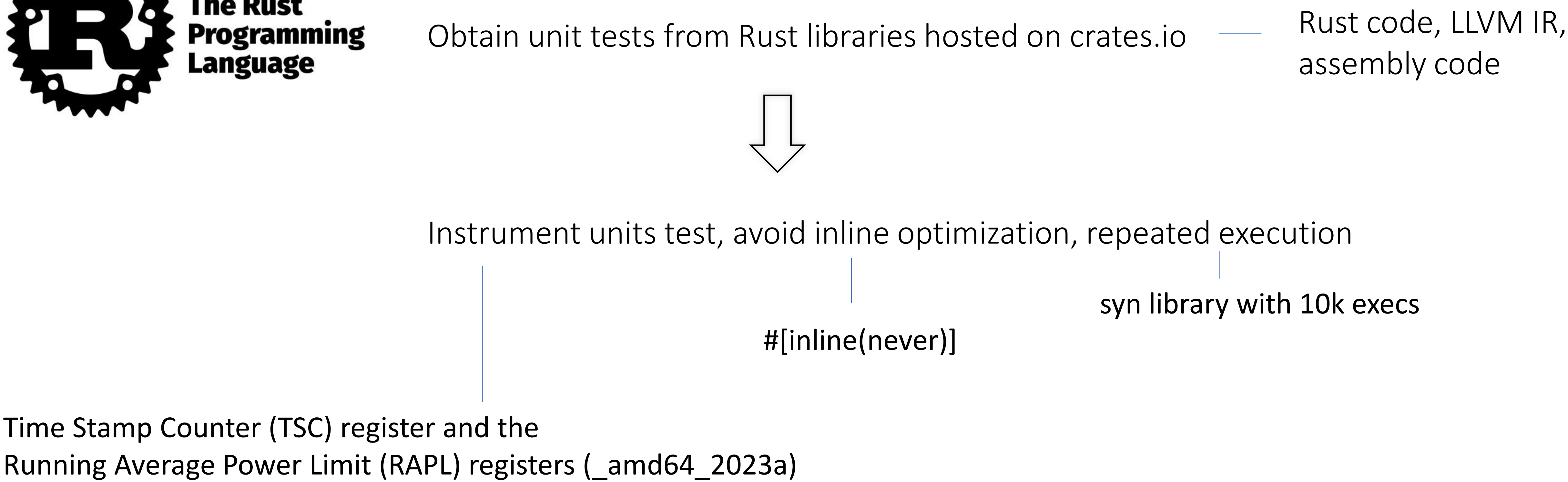
Valid Assumptions

- Code, intermediate representation, and assembler code map to processor instructions that require different amounts of time
- Execution time might correspond to energy consumed (more or less)

- Using assembler might even account for compiler optimization

- If you know the impact of code on different levels of abstraction on energy consumption you might be able to generalize

Data Collection



Obtain unit tests from Rust libraries hosted on crates.io

Rust code, LLVM IR, assembly code

Instrument units test, avoid inline optimization, repeated execution

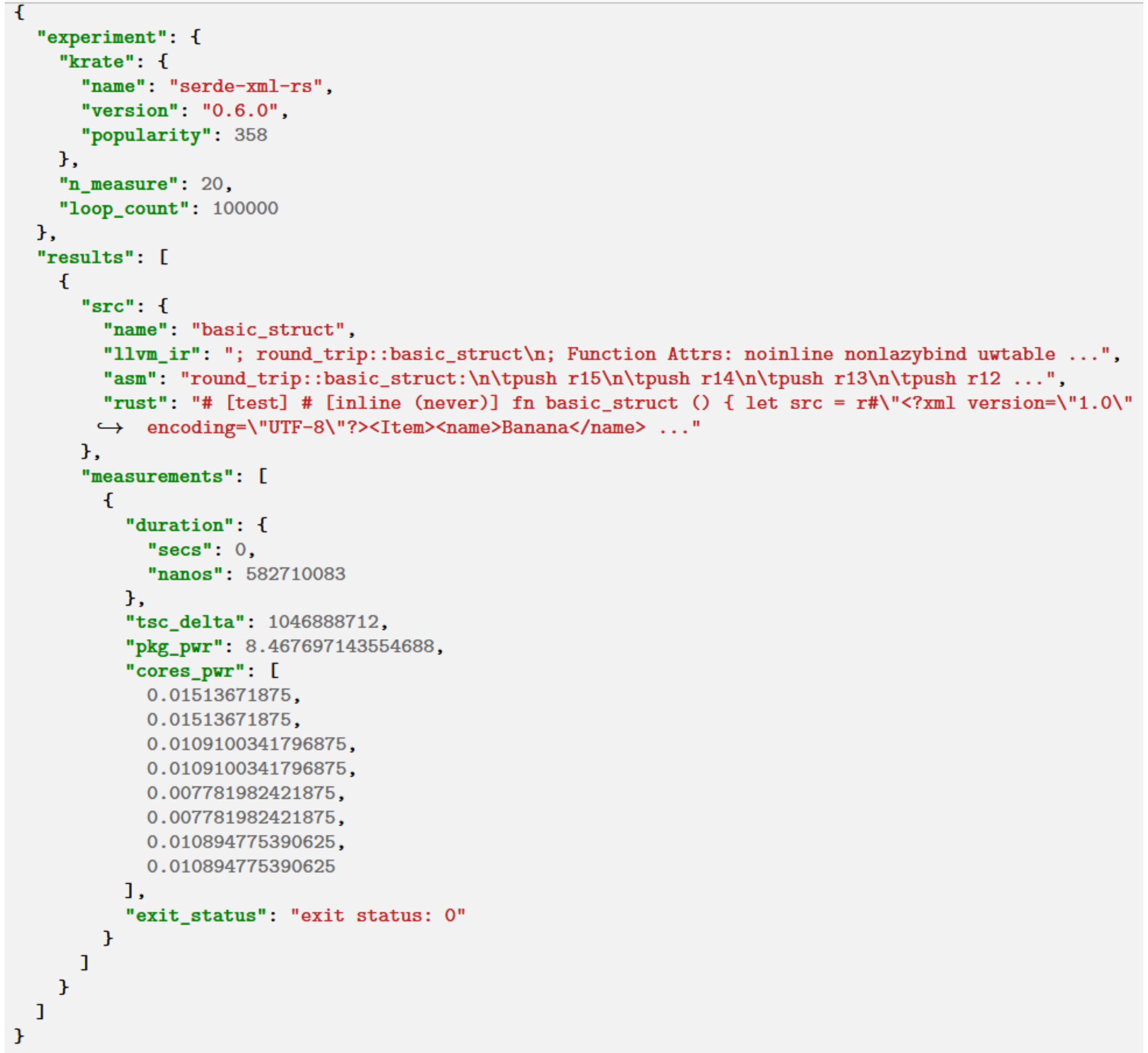
syn library with 10k execs

#[inline(never)]

Time Stamp Counter (TSC) register and the
Running Average Power Limit (RAPL) registers (_amd64_2023a)

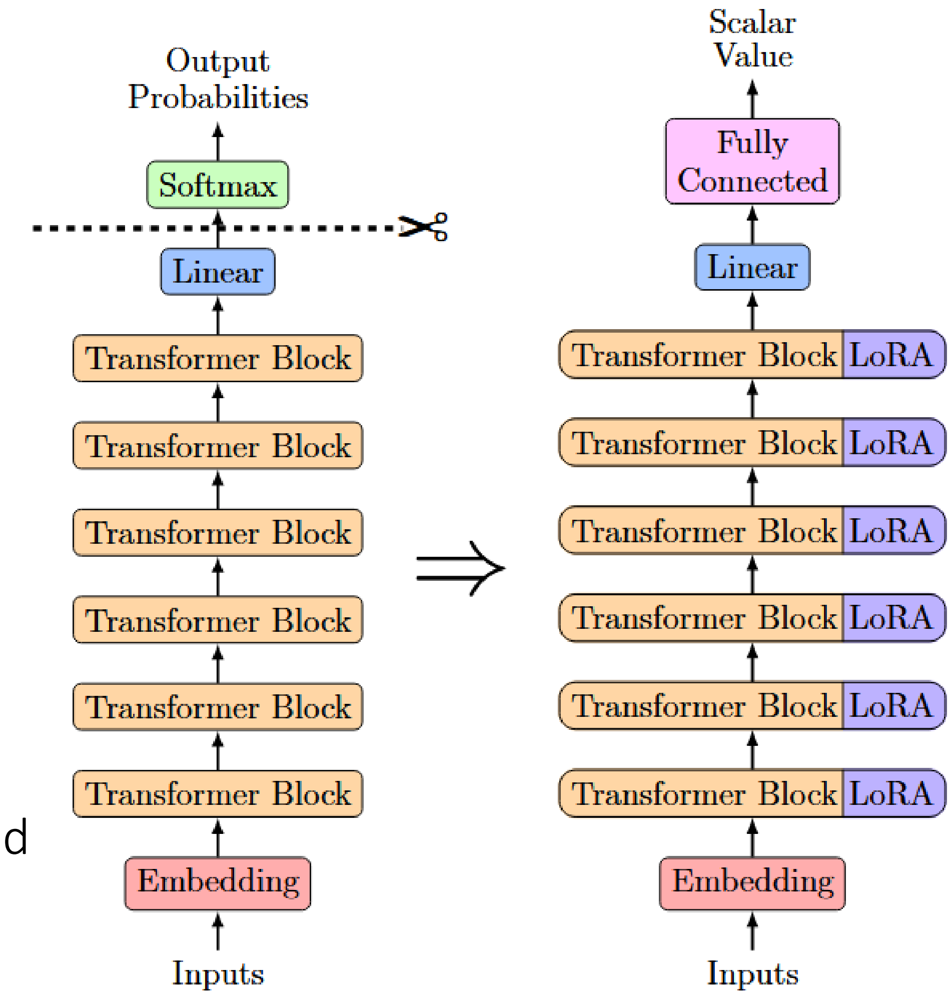
Total Effort

- Repeat measurements 20x
- 4400 unit tests measuremed from the top 1000 most popular Rust crates

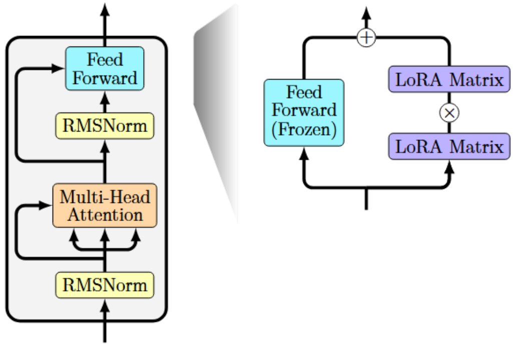


```
{
  "experiment": {
    "krate": {
      "name": "serde-xml-rs",
      "version": "0.6.0",
      "popularity": 358
    },
    "n_measure": 20,
    "loop_count": 100000
  },
  "results": [
    {
      "src": {
        "name": "basic_struct",
        "llvm_ir": "; round_trip::basic_struct\n; Function Attrs: noinline nonlazybind uwtable ...",
        "asm": "round_trip::basic_struct:\n\tpush r15\n\tpush r14\n\tpush r13\n\tpush r12 ...",
        "rust": "# [test] # [inline (never)] fn basic_struct () { let src = r#\"<?xml version=\"1.0\"
       ↪  encoding=\"UTF-8\"?><Item><name>Banana</name> ..."
      },
      "measurements": [
        {
          "duration": {
            "secs": 0,
            "nanos": 582710083
          },
          "tsc_delta": 1046888712,
          "pkg_pwr": 8.467697143554688,
          "cores_pwr": [
            0.01513671875,
            0.01513671875,
            0.0109100341796875,
            0.0109100341796875,
            0.007781982421875,
            0.007781982421875,
            0.010894775390625,
            0.010894775390625
          ],
          "exit_status": "exit status: 0"
        }
      ]
    }
  ]
}
```

LLM Adaptation and Fine-Tuning



1. LoRa adaptation to reduce computational effort



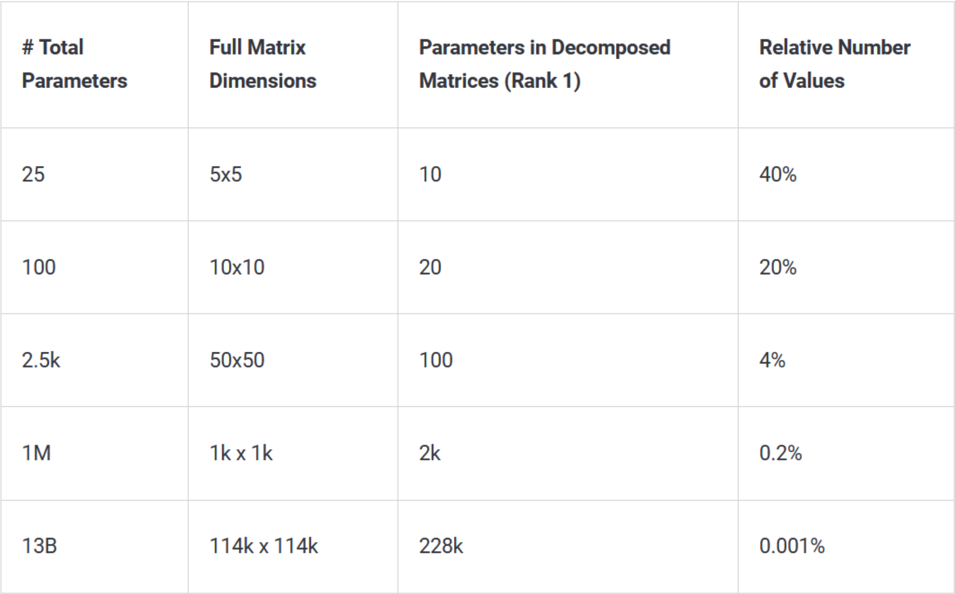
2. Replacement of the last layer with a new fully connected layer that outputs a scalar number rather than token probabilities.

LoRa Excursion

Idea:

- Track only the delta in parameter (i.e., changes that would be made to the original weights) that stem from the new data
- Decompose the large weight matrix into smaller matrices that contain only the parameters to be trained

$$\begin{bmatrix} 1 \\ 3 \\ 7 \\ -4 \\ 2 \end{bmatrix} \times \begin{bmatrix} 5 & 1 & -1 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 5 & 1 & -1 & 3 & 4 \\ 15 & 3 & -3 & 9 & 12 \\ 35 & 7 & -7 & 21 & 28 \\ -20 & -4 & 4 & -12 & -16 \\ 10 & 2 & -2 & 6 & 8 \end{bmatrix}$$

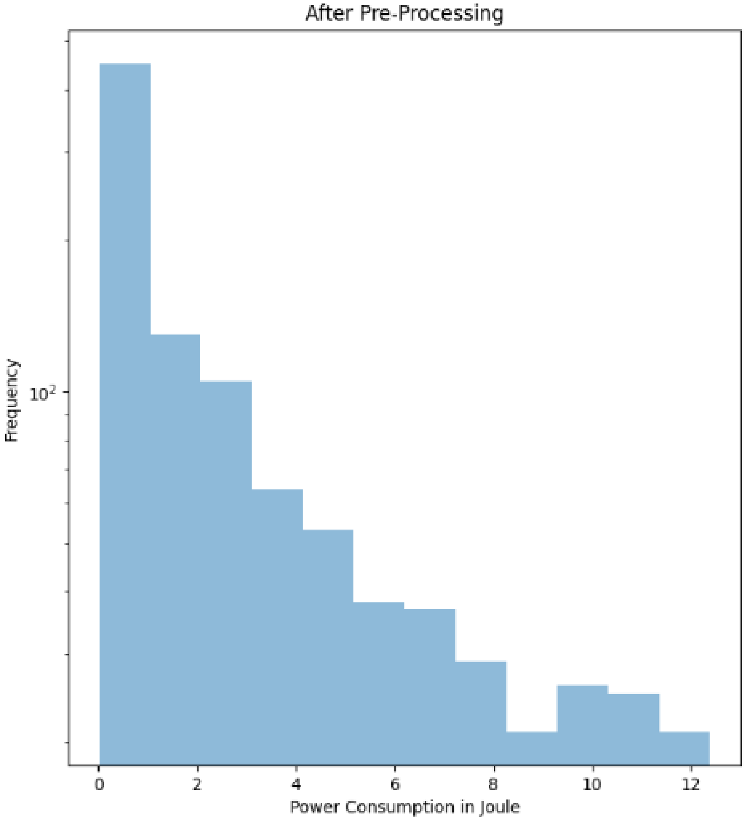


# Total Parameters	Full Matrix Dimensions	Parameters in Decomposed Matrices (Rank 1)	Relative Number of Values
25	5x5	10	40%
100	10x10	20	20%
2.5k	50x50	100	4%
1M	1k x 1k	2k	0.2%
13B	114k x 114k	228k	0.001%

Results

Majority of measurements are below 1J or 2J, which makes learning complicated.

Bag of Words has lowest error and the LLM performs worse than a simple linear model


After Pre-Processing

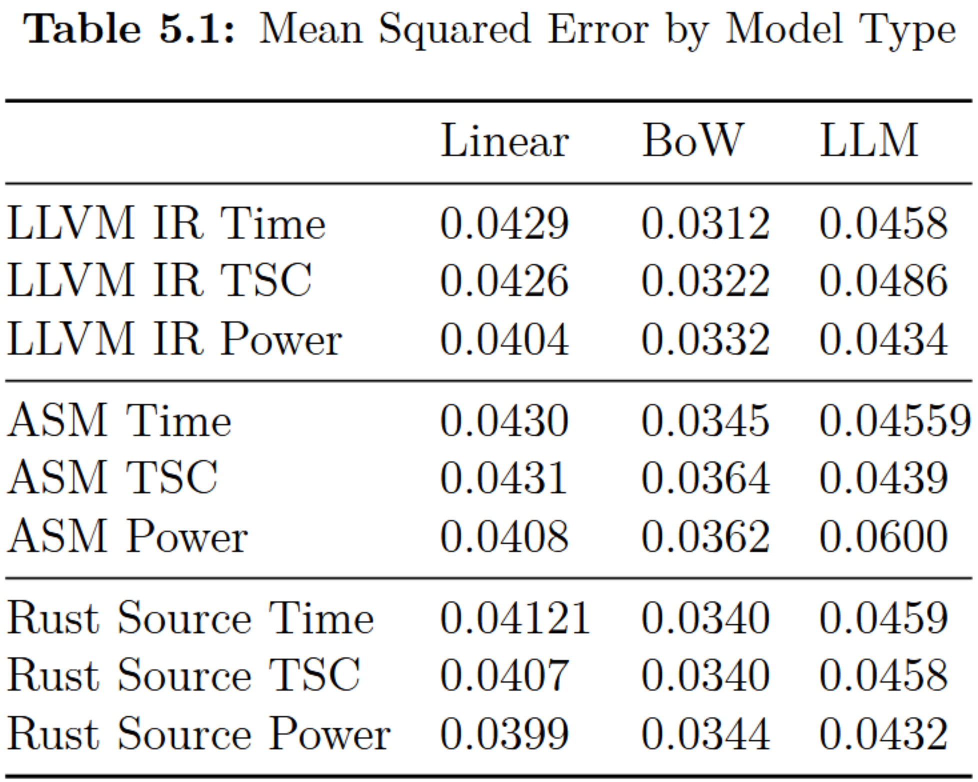


Table 5.1: Mean Squared Error by Model Type

	Linear	BoW	LLM
LLVM IR Time	0.0429	0.0312	0.0458
LLVM IR TSC	0.0426	0.0322	0.0486
LLVM IR Power	0.0404	0.0332	0.0434
ASM Time	0.0430	0.0345	0.04559
ASM TSC	0.0431	0.0364	0.0439
ASM Power	0.0408	0.0362	0.0600
Rust Source Time	0.04121	0.0340	0.0459
Rust Source TSC	0.0407	0.0340	0.0458
Rust Source Power	0.0399	0.0344	0.0432

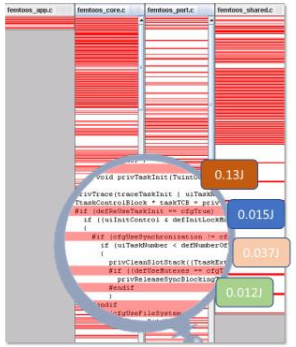
Discussion: Why????

Too few training examples: very likely

Unclear relationship between code and energy consumption: very likely

No information about control- and data-flow: very likely

The Dream



```
  void privTaskInit(TuintO
rivTrace(traceTaskInit | uiTask
TtaskControlBlock * taskTCB = priv
#if (defReUseTaskInit == cfgTrue)
  if ((uiInitControl & defInitLockM
  {
    #if (cfgUseSynchronization != cf
      if (uiTaskNumber < defNumberOf
      {
        privCleanSlotStack((TtaskExt
        #if ((defUseMutexes == cfgT
          privReleaseSyncBlockingT
        #endif
      }
    endif
    cfgUseFileSystem
```

How? You have read the title… right? ☺

The Idea

LLMs are great for code understanding, generation, and summarization

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.scre
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate,'%a %b %d %H:%M:%S +0000 %Y'
        today = datetime.now()
        howlong = (today-ltdate2).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past' , daywindow,
            totaltweets += len(response.data)
            for j in response.data:
                if j.entities.urls:
                    for k in j.entities.urls:
                        newurl = k['expanded_url']
                        urlset.add((newurl, j.user.screen_name))
        else:
            print i.screen_name, 'has not tweeted in the past', daywind
```

⇨

⇨

Can we adapt and fine-tune an LLM to estimate energy consumption of given code?

Thanks!

Data Collection

The Rust Programming Language

Obtain unit tests from Rust libraries hosted on crates.io — Rust code, LLVM IR, assembly code

⇩

Instrument units test, avoid inline optimization, repeated execution

Time Stamp Counter (TSC) register and the Running Average Power Limit (RAPL) registers (_amd64_2023a)

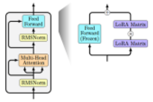
#[inline(never)]

syn library with 10k execs

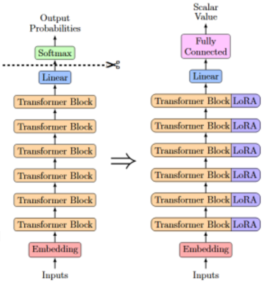
LLM Adaptation and Fine-Tuning

LLaMA 2

1. LoRa adaptation to reduce computational effort

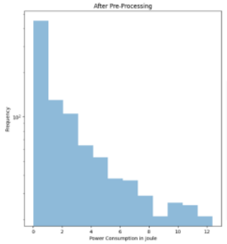


2. Replacement of the last layer with a new fully connected layer that outputs a scalar number rather than token probabilities.



Results

Majority of measurements are below 1J or 2J, which makes learning complicated.



Bag of Words has lowest error and the LLM performs worse that a simple linear model

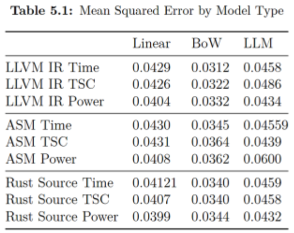


Table 5.1: Mean Squared Error by Model Type

	Linear	BoW	LLM
LLVM IR Time	0.0429	0.0312	0.0458
LLVM IR TSC	0.0426	0.0322	0.0486
LLVM IR Power	0.0404	0.0332	0.0434
ASM Time	0.0430	0.0345	0.04559
ASM TSC	0.0431	0.0364	0.0439
ASM Power	0.0408	0.0362	0.0600
Rust Source Time	0.04121	0.0340	0.0459
Rust Source TSC	0.0407	0.0340	0.0458
Rust Source Power	0.0399	0.0344	0.0432