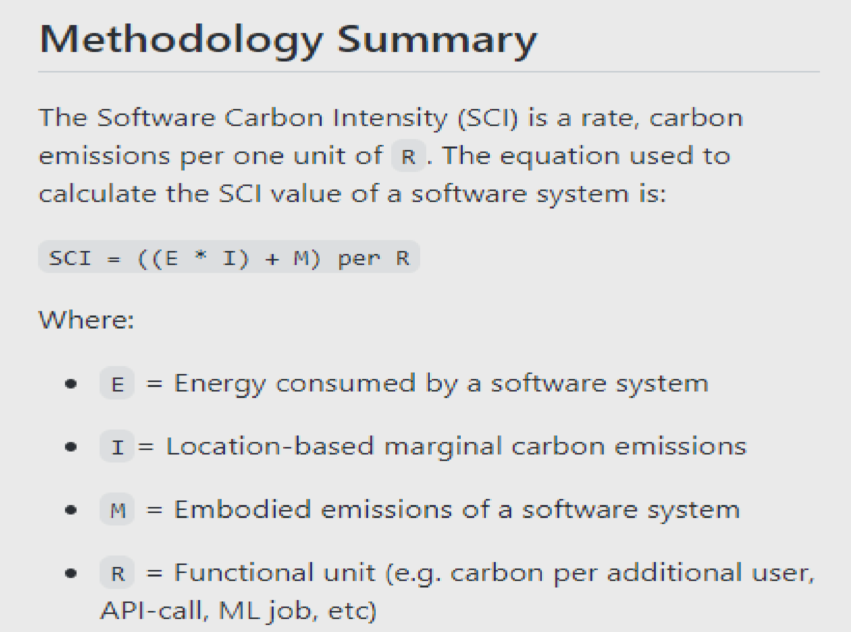
Software Decarbonization & Programming Languages Role On Social Cost of Carbon

Chris Xie
Chris.Xie@futurewei.com
January 2023

GSF SCI Gaps: Benchmarks and SCI Scores



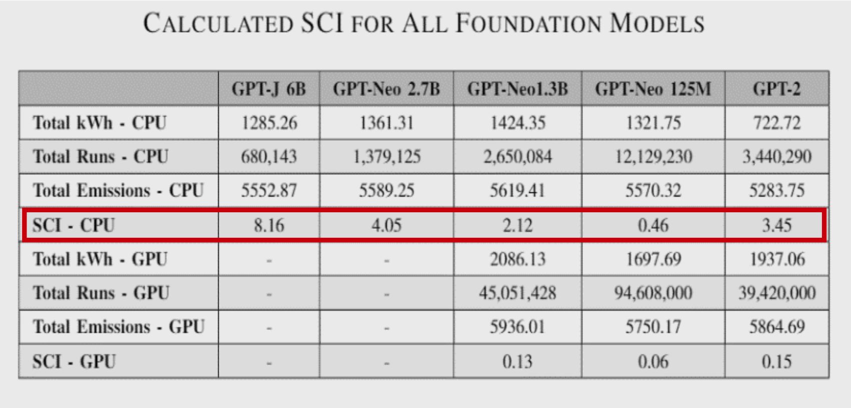
Methodology Summary

The Software Carbon Intensity (SCI) is a rate, carbon emissions per one unit of `R`. The equation used to calculate the SCI value of a software system is:

```
SCI = ((E * I) + M) per R
```

Where:

- `E` = Energy consumed by a software system
- `I` = Location-based marginal carbon emissions
- `M` = Embodied emissions of a software system
- `R` = Functional unit (e.g. carbon per additional user, API-call, ML job, etc)

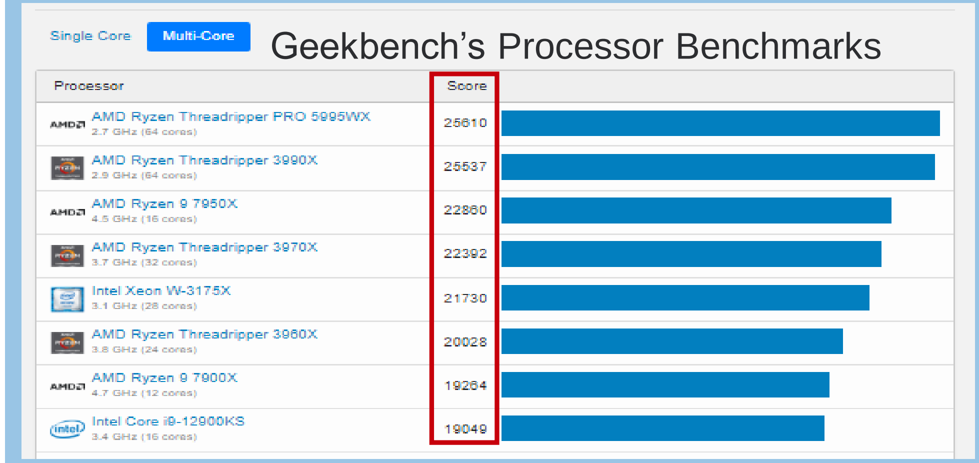


Calculated SCI for All Foundation Models

	GPT-J 6B	GPT-Neo 2.7B	GPT-Neo1.3B	GPT-Neo 125M	GPT-2
Total kWh - CPU	1285.26	1361.31	1424.35	1321.75	722.72
Total Runs - CPU	680,143	1,379,125	2,650,084	12,129,230	3,440,290
Total Emissions - CPU	5552.87	5589.25	5619.41	5570.32	5283.75
SCI - CPU	8.16	4.05	2.12	0.46	3.45
Total kWh - GPU	-	-	2086.13	1697.69	1937.06
Total Runs - GPU	-	-	45,051,428	94,608,000	39,420,000
Total Emissions - GPU	-	-	5936.01	5750.17	5864.69
SCI - GPU	-	-	0.13	0.06	0.15

Gaps

- Define a benchmark, workload, demo, referencing MLPerf/Geekbench benchmarks
- Use cases to demonstrate business value of SCI
- Value propositions of SCI : correlate SCI scores to dollar amount of carbon emissions of software systems



Geekbench's Processor Benchmarks

Single Core

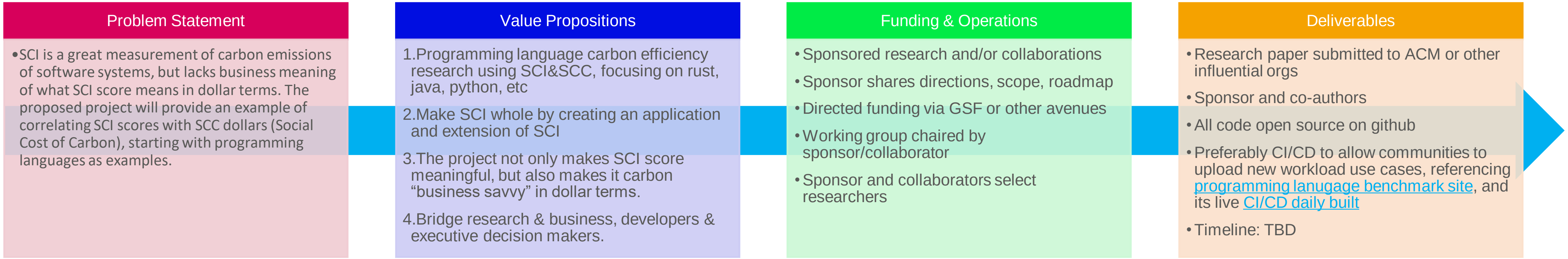
Multi-Core

Processor	Score
AMD Ryzen Threadripper PRO 5995WX 2.7 GHz (64 cores)	25610
AMD Ryzen Threadripper 3990X 2.9 GHz (64 cores)	25537
AMD Ryzen 9 7950X 4.5 GHz (16 cores)	22860
AMD Ryzen Threadripper 3970X 3.7 GHz (32 cores)	22392
Intel Xeon W-3175X 3.1 GHz (28 cores)	21730
AMD Ryzen Threadripper 3960X 3.8 GHz (24 cores)	20028
AMD Ryzen 9 7900X 4.7 GHz (12 cores)	19264
Intel Core i9-12900KS 3.4 GHz (16 cores)	19049

From the community: "…the introduction of a basic 'working example' inline in the spec itself, would also greatly help with readability and understanding of SCI…"

SCI Opportunities: SCI-based Social Cost of Carbon (SCC) Study on Programming Languages & Software Systems

Social Cost of Carbon (SCC) is an estimate, in monetary value, the economic damages that would result from emitting one additional ton of greenhouse gases into the atmosphere, in a given year. A multi-year study finds that per ton CO2 emitted into the atmosphere costs society $185.



Problem Statement

- SCI is a great measurement of carbon emissions of software systems, but lacks business meaning of what SCI score means in dollar terms. The proposed project will provide an example of correlating SCI scores with SCC dollars (Social Cost of Carbon), starting with programming languages as examples.

Value Propositions

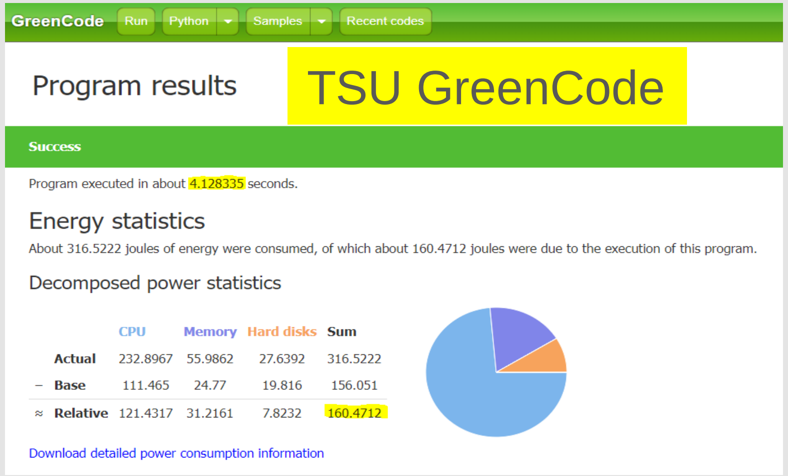
1. Programming language carbon efficiency research using SCI&SCC, focusing on rust, java, python, etc
2. Make SCI whole by creating an application and extension of SCI
3. The project not only makes SCI score meaningful, but also makes it carbon "business savvy" in dollar terms.
4. Bridge research & business, developers & executive decision makers.

Funding & Operations

- Sponsored research and/or collaborations
- Sponsor shares directions, scope, roadmap
- Directed funding via GSF or other avenues
- Working group chaired by sponsor/collaborator
- Sponsor and collaborators select researchers

Deliverables

- Research paper submitted to ACM or other influential orgs
- Sponsor and co-authors
- All code open source on github
- Preferably CI/CD to allow communities to upload new workload use cases, referencing programming lanugage benchmark site, and its live CI/CD daily built
- Timeline: TBD



Program results

TSU GreenCode

Success

Program executed in about 4.128335 seconds.

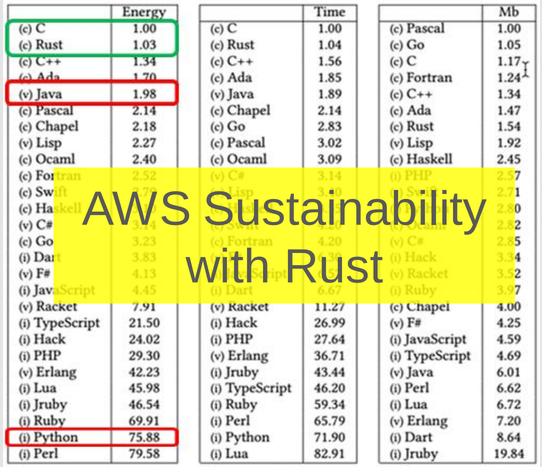
Energy statistics

About 316.5222 joules of energy were consumed, of which about 160.4712 joules were due to the execution of this program.

Decomposed power statistics

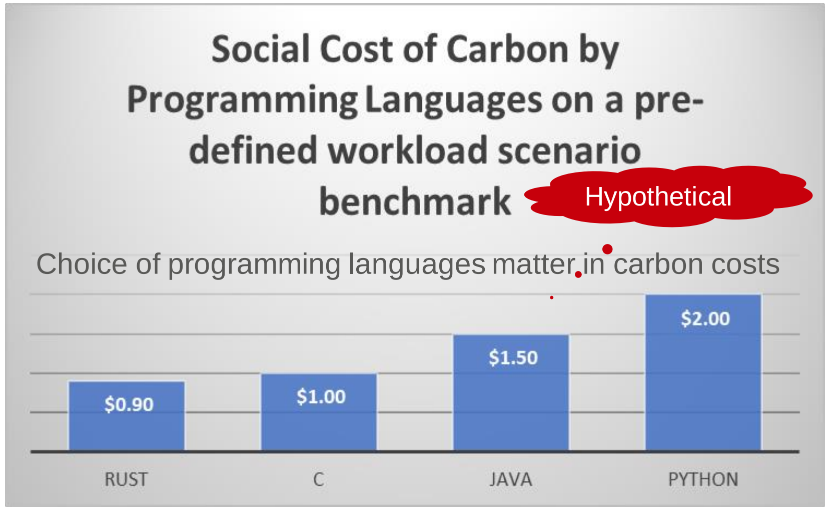
		CPU	Memory	Hard disks	Sum
	Actual	232.8967	55.9862	27.6392	316.5222
−	Base	111.465	24.77	19.816	156.051
≈	Relative	121.4317	31.2161	7.8232	160.4712

Download detailed power consumption information



	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

AWS Sustainability with Rust



Social Cost of Carbon by Programming Languages on a pre-defined workload scenario benchmark

Hypothetical

Choice of programming languages matter in carbon costs

Thank You!