A Study in Concurrency Benchmarking

Simon Charalambous, Jordan Loh, Yerzhan Mazhkenov, Karolis Mituzas, Ethan Moss, Andreas Voskou

Supervisors:
Prof. Sophia Drossopoulou, Juliana Franco

A MSC COMPUTING SCIENCE GROUP PROJECT
IMPERIAL COLLEGE LONDON
2017

Goal

Compare performance of 3 languages with concurrency capabilities and focus:

- Pony
- Scala (with Akka actor Library)
- Go

Outcomes

Wrote 3 Benchmarks

- Ping Pong Test
- Sleeping Barber Problem
- N-Body Simulation

Developed a front-end benchmarking tool in Python

Analysed benchmarking results

Made observations on the languages

Background on Concurrency

Slowing gains in sequential performance speed

Programs today…

- Run on many cores/threads
- Used by many users simultaneously
- Need to write them in a **concurrent-safe manner**

However, “low-level” concurrency is not easy

- Use of mutexes, locks, semaphores, monitors
- Can result in deadlocks and data-races if not careful

Modern approaches

Actor Model (Hewitt et al. 1973)

- Everything' is an actor
- Actors send messages to each other **asynchronously**
- Messages stored in actors' mailboxes until processed
- Languages: **Scala (Akka), Pony**, Erlang, etc.

Communicating Sequential Processes (Hoare 1978)

- Processes use explicit channels for message-passing
- Involves a rendezvous between processes:
  - i.e. the sender cannot transmit a message until the receiver is ready to accept it
- Languages: **Go**, Ada, etc.

Modern approaches (cont.)

Both Actor and CSP models are ‘higher level’ models

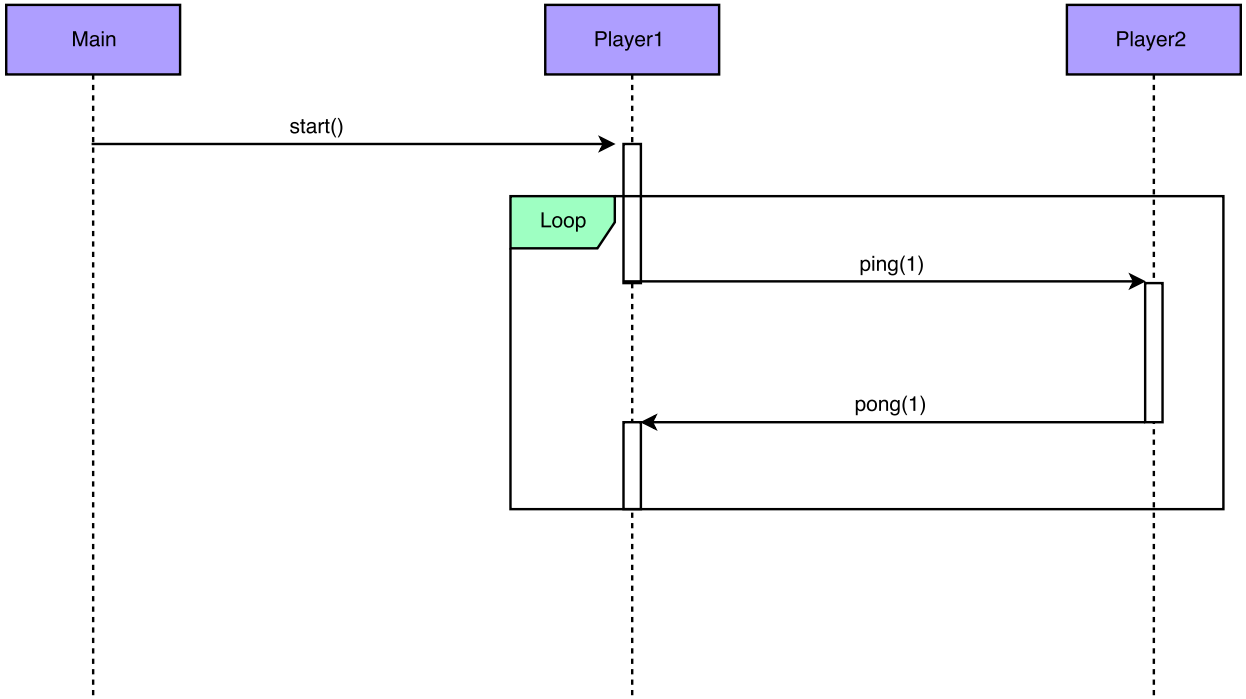
- Involve message passing
- Abstract away manually handling mutexes, locks, semaphores, monitors
- -> Easier concurrency programming!

For the 3 languages we have chosen (Scala, Pony, and Go)

- How do they perform in terms of speed and memory utilisation?
- How do we test their concurrency features? (Benchmarks)
- How easy is it to learn and program in them?

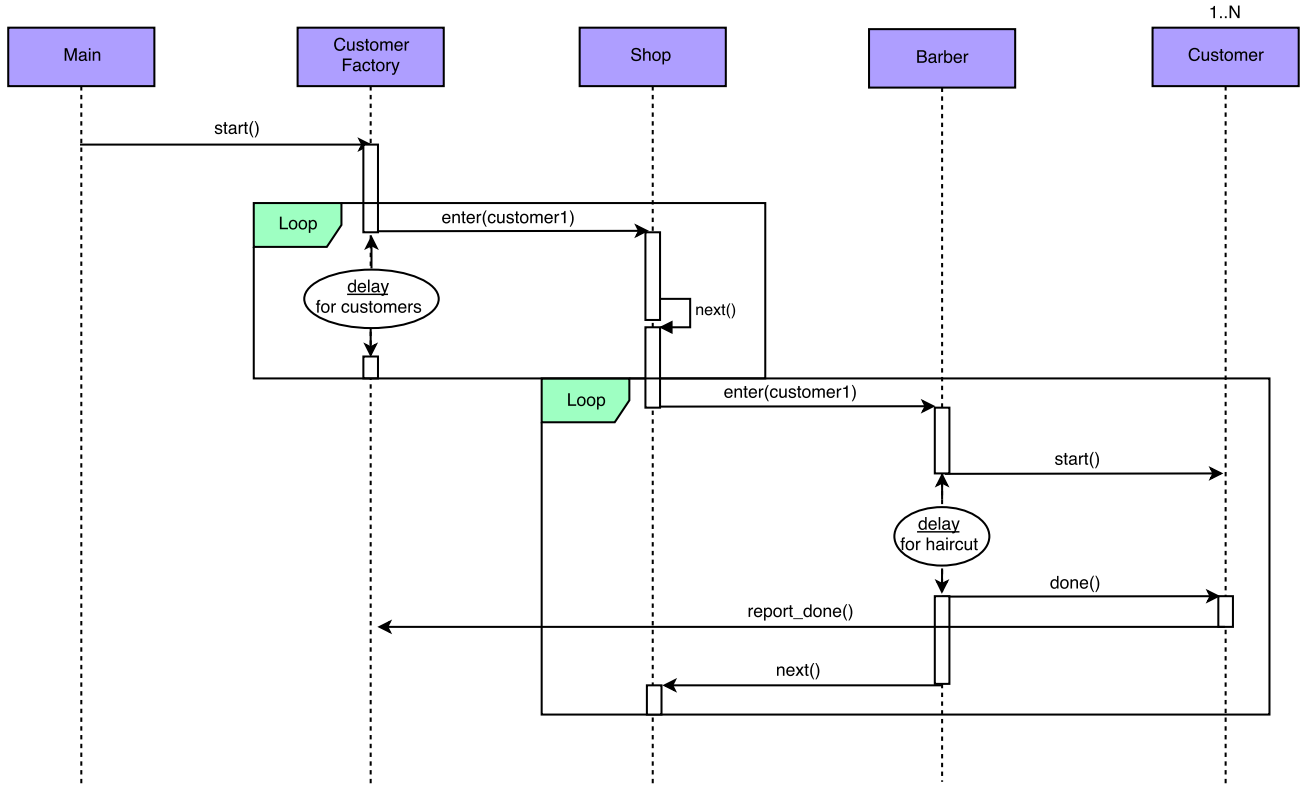
Benchmark Design: Ping Pong Test

- Two concurrent entities exchanging messages
- Tests message passing performance



Benchmark Design: Sleeping Barber Problem

- Tests communication and synchronization



Benchmark Design: N-Body Simulation

- N bodies interacting with each other gravitationally
- Equation of the motion (for each body):

$$m_i \frac{d^2 \mathbf{r}_i}{dt^2} = \sum_{\substack{j=1 \\ j \neq i}}^{N} \boldsymbol{\rho}_{ij} \frac{m_1 m_2 G}{|\boldsymbol{\rho}_{ij}|^3}, \quad \boldsymbol{\rho}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$$

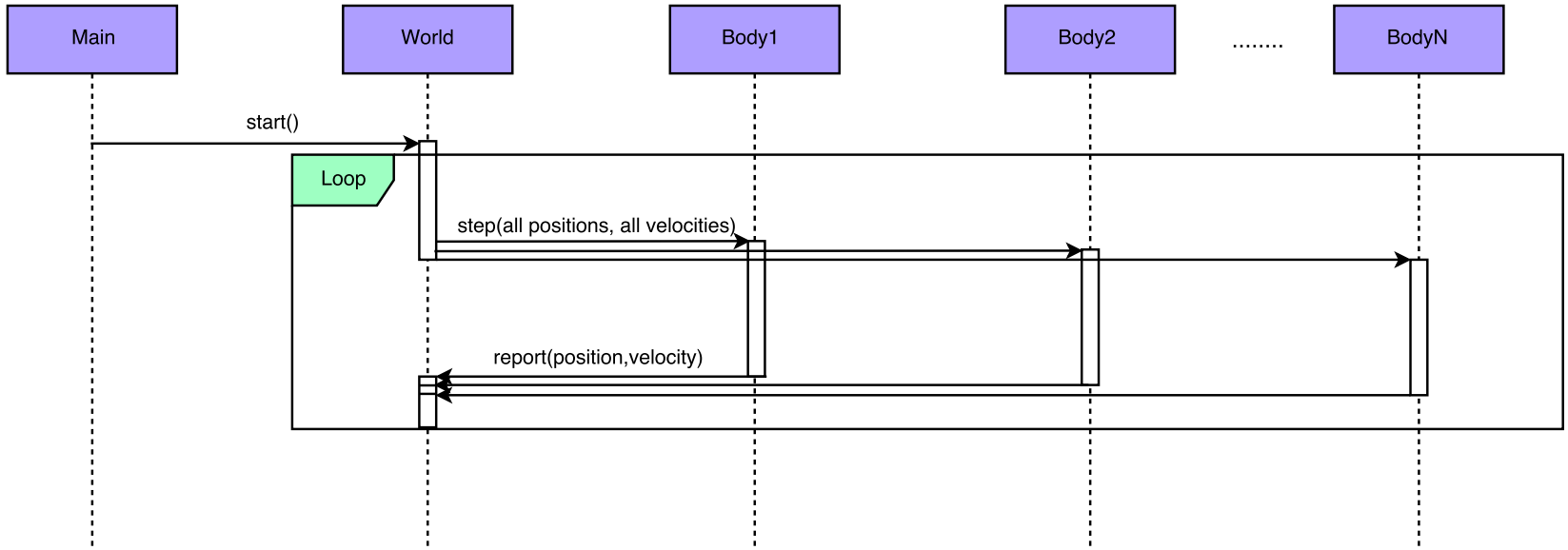
- Parallel numerical integration of the equations
- Direct gravitational N-body simulation
  - Calculate all the forces
  - No simplifications

Tests the performance of

- Parallel calculations
- Message passing
- Handling multiple actors/goroutines

Benchmark Design: N-Body Simulation (cont.)

- Each body is an Actor (or goroutine)
- Bodies calculates their next position concurrently



Front-end Tool

Tedious to run benchmarks manually

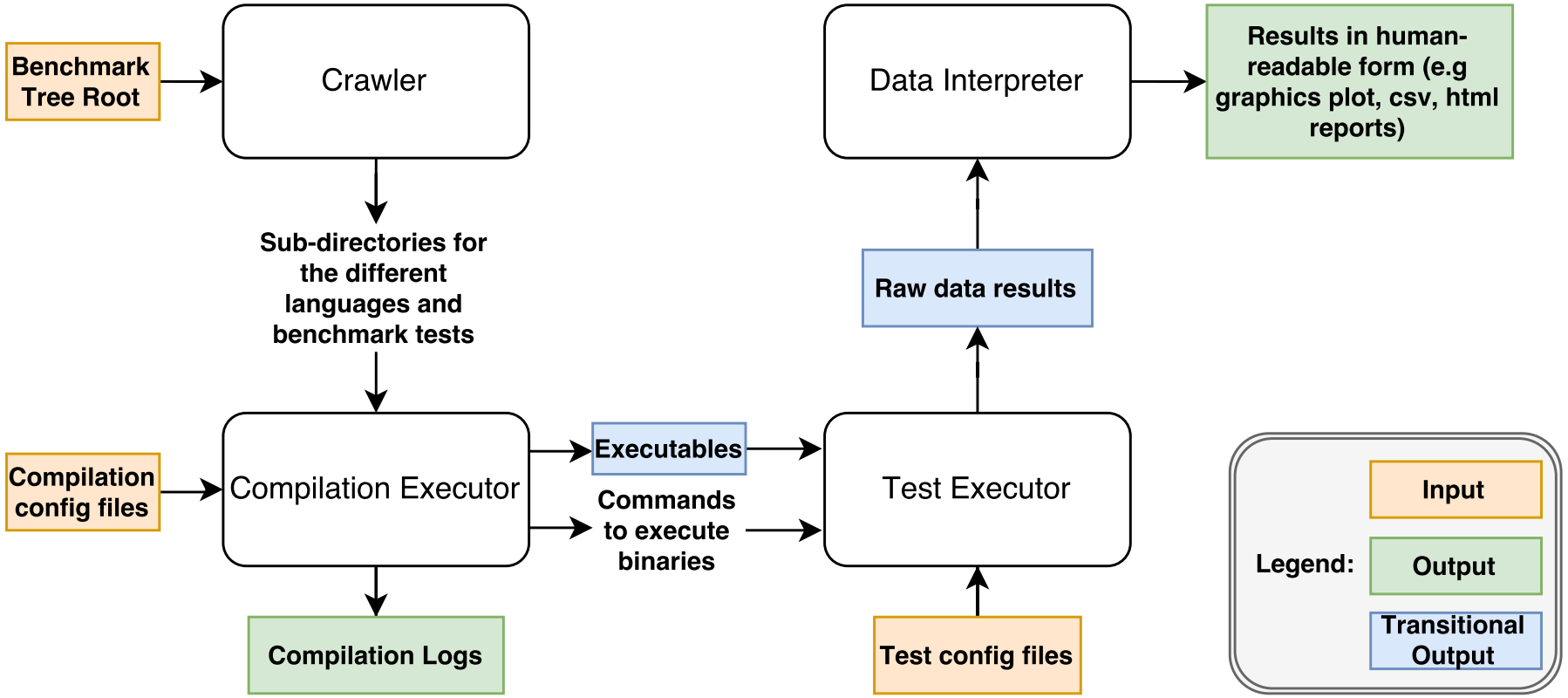
UNIX time utility

- Time, memory, CPU usage etc.

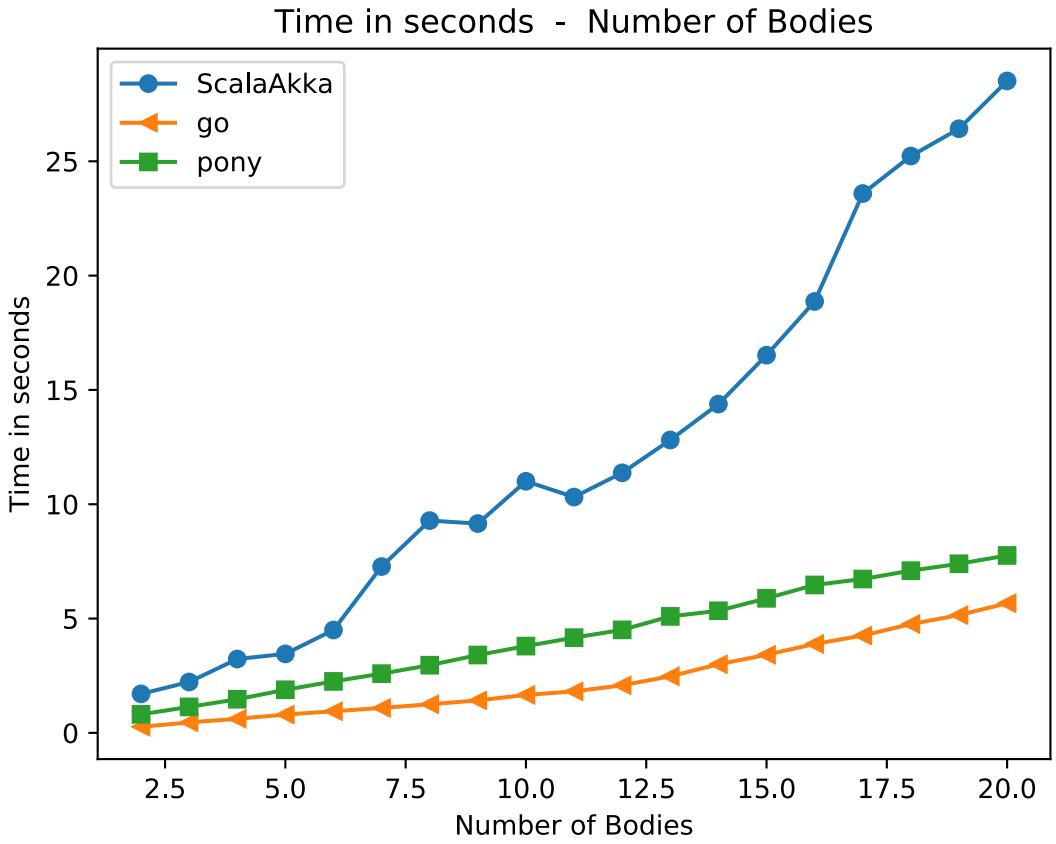
Python Script

Extensibility

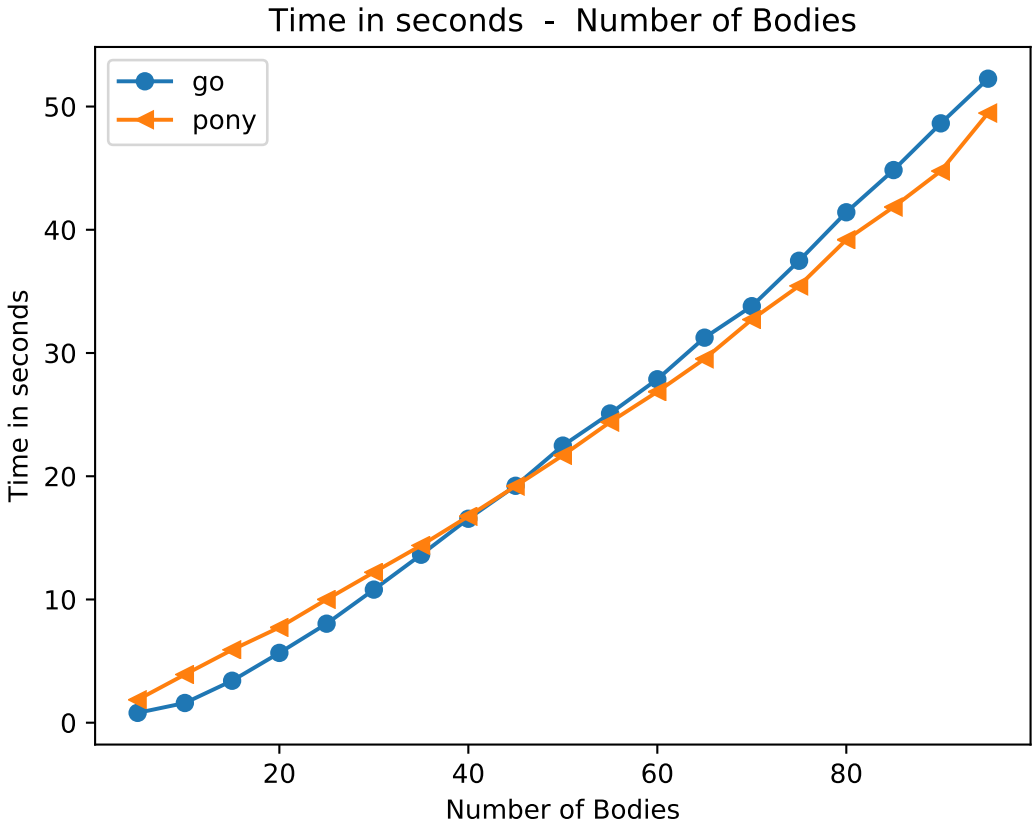
Front-end Tool: Structure and Demonstration



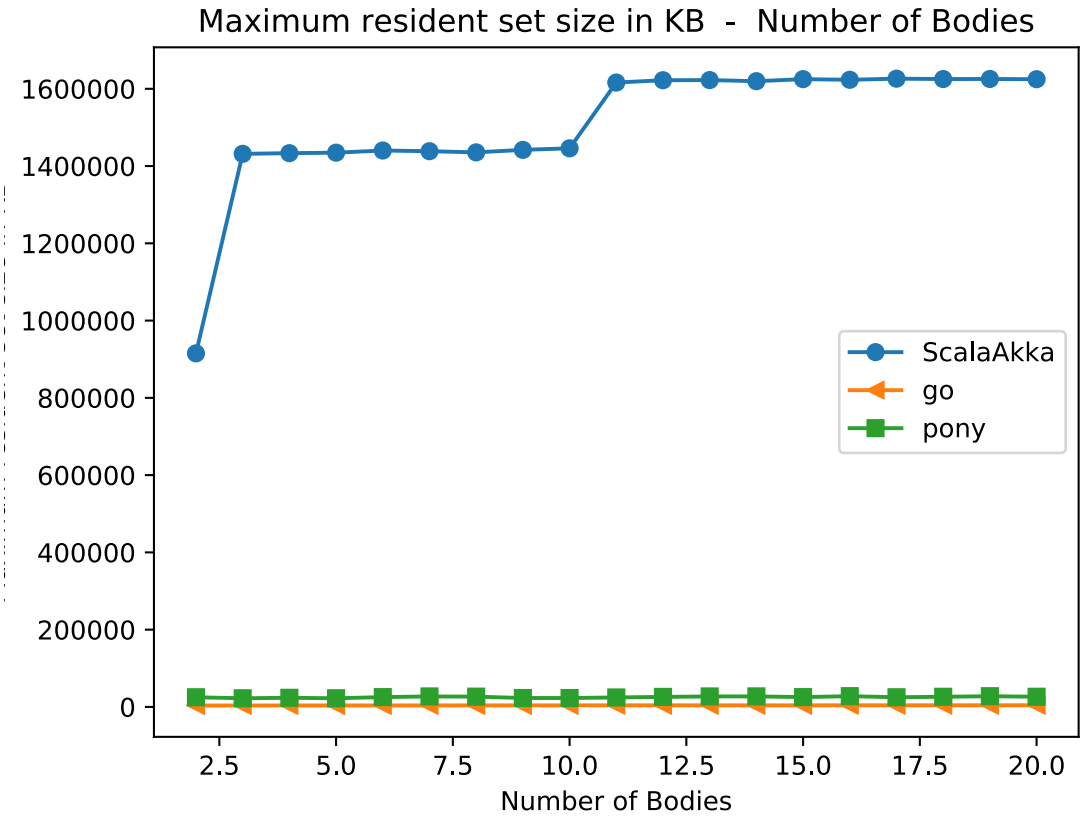
N-Body Results: Runtime



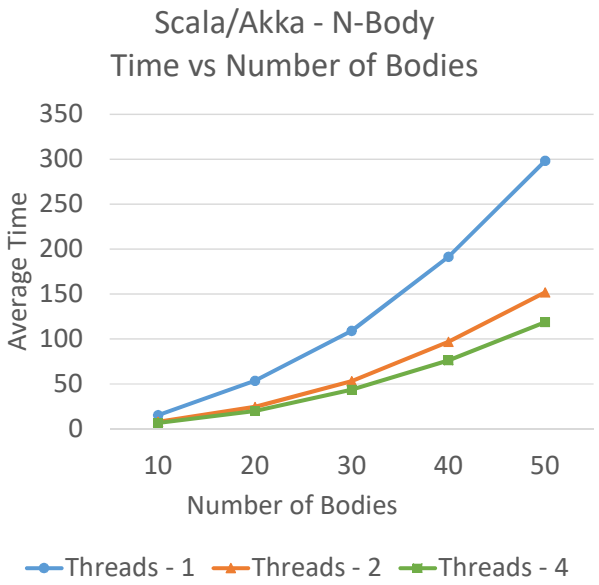
N-Body Results: Runtime (cont.) – more bodies

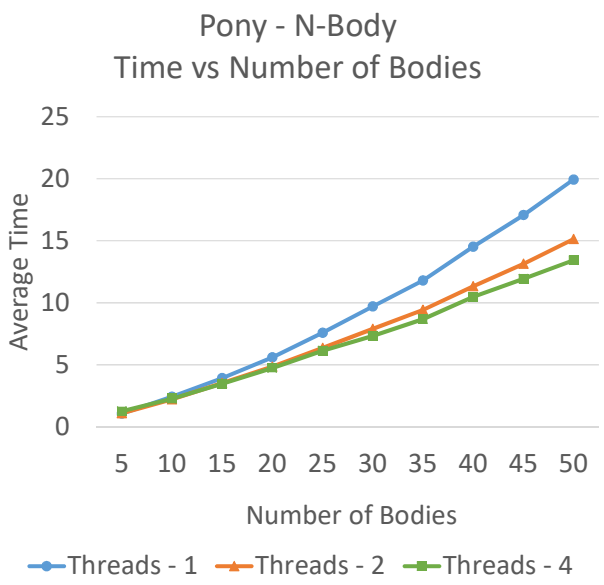


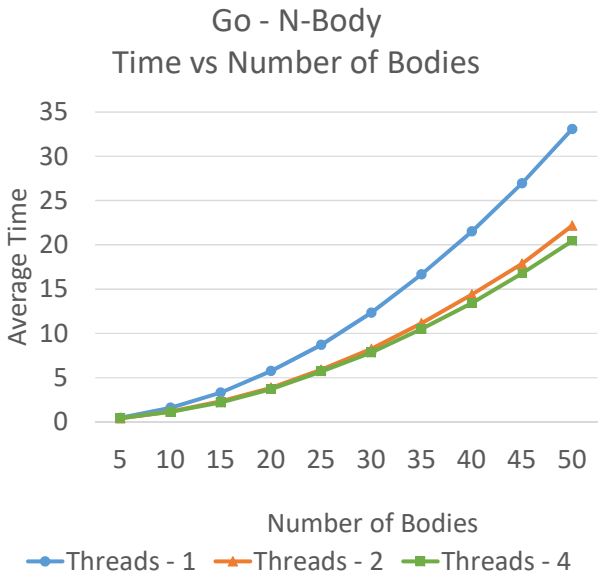
N-Body Results: Peak Memory Usage



N-Body Results: CPU Thread Scaling







Observations on Languages

- All members wrote code across all 3 benchmark languages
- Actual lines of code required for the benchmarks lower than expected
  - Expressive power of the languages -> succinct concurrent programs
  - Remember: no need to explicitly write mutexes, locks, semaphores etc.
- Learning to code in the Actor/CSP paradigms, and ensuring consistency of benchmark implementation across languages not easy and took time

Lines of code	Ping Pong	Barber	N-Body	Total
Pony	70	172	282	524
Go	60	279	322	661
Scala/Akka	77	223	307	607
Benchmarks	*207*	*674*	*911*	*1792*

Python: Front-End Tool = 918, Animation = 137, Total = 1055

Observations on Languages

Scala

- like Java; lots of syntactic sugar; OOP+Functional Programming, powerful actor library (Akka)
- but…slow; using build tools like SBT not straightforward

Pony

- unique security capability features provide safety in concurrency programming; fast; supportive developer community
- but…developing; limited 3rd-party learning resources; not as popular as other 2 languages; steep learning curve

Go

- simplicity; popular language with wide-scale adoption and support; fast
- implementing CSP/channels sometimes feels less elegant than actor-model solutions

Conclusion

Representativeness of benchmark

- Confident in results of N-Body and Ping Pong
- Sleeping Barber showed unexpected results with anomalies
  - Possibly caused by the use of “random timers” to simulate length of a haircut

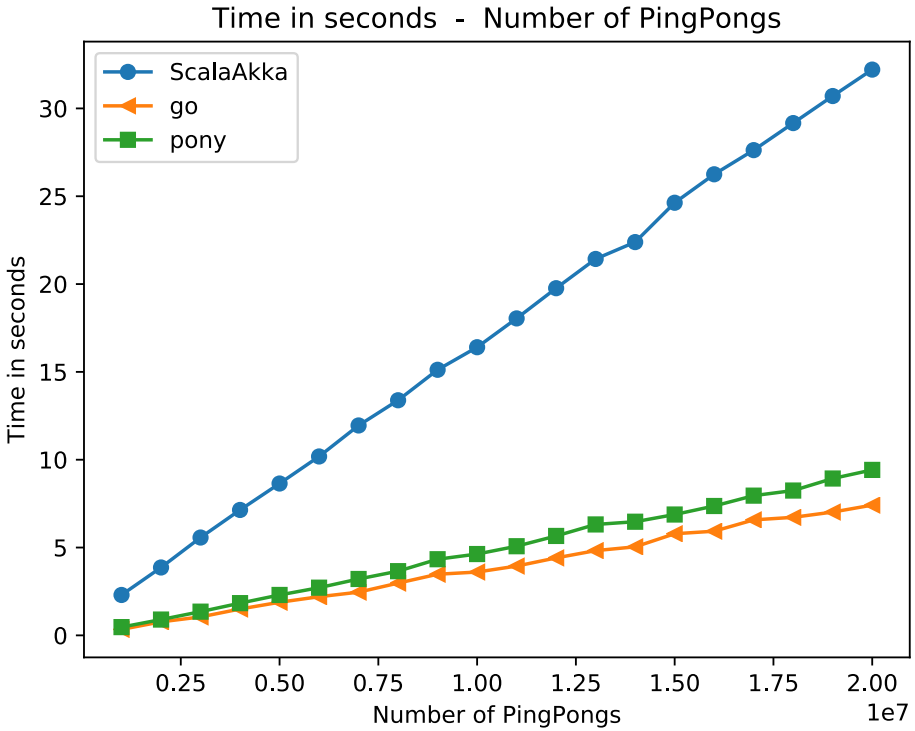
Front-end tool usefulness

- Auto-generation of comparison diagrams
- All raw data (time, memory, switches etc.) generated by tool
- Automated detection of new languages and benchmarks
  - Provided python config files are written properly by the user
- Tested on Ubuntu, Slackware Linux, and MacOS Sierra

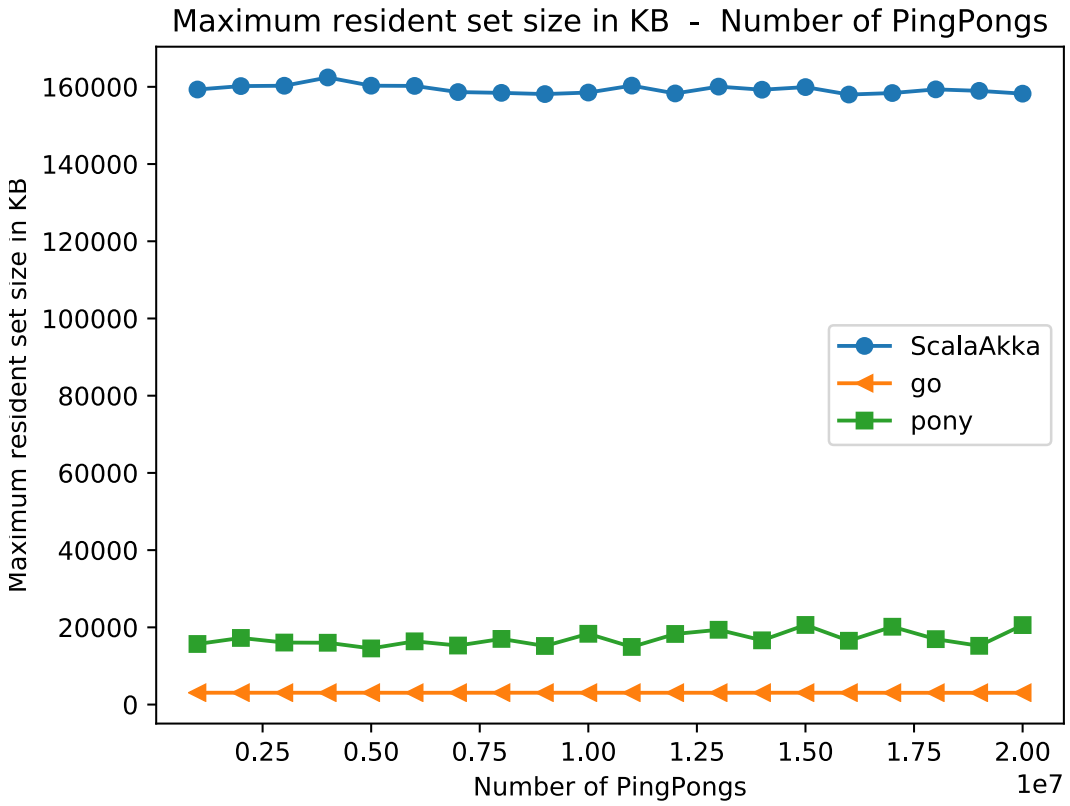
Thank you!
Q&A

Appendix

Ping Pong Results: Runtime

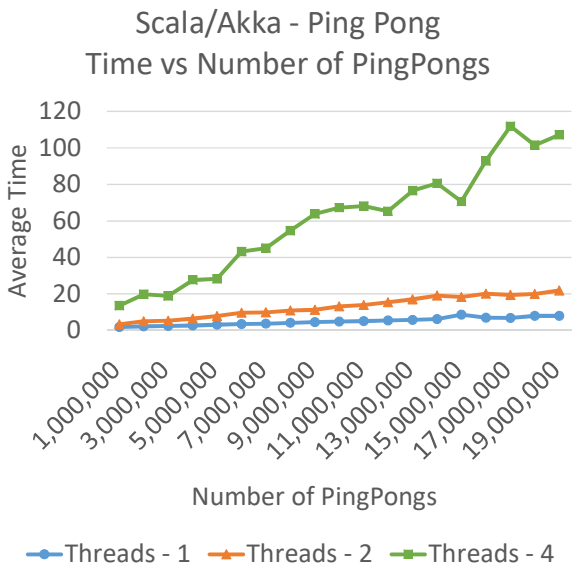


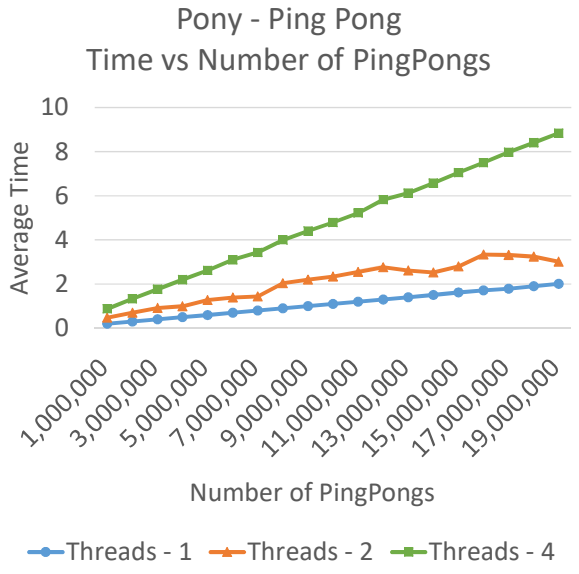
Ping Pong Results: Peak Memory Usage

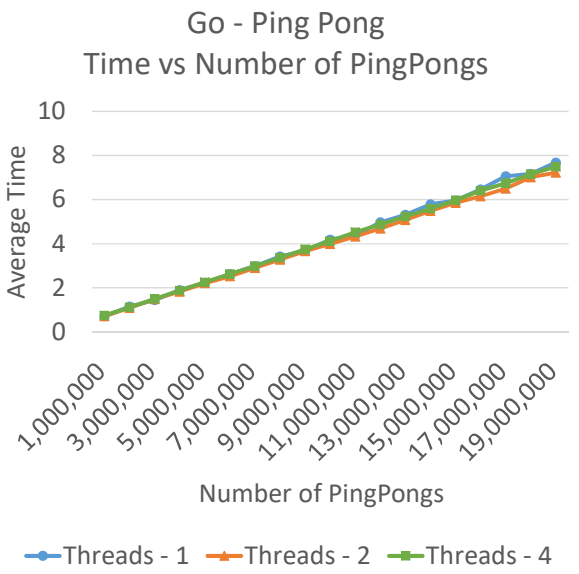


Ping Pong Results: CPU Thread Scaling

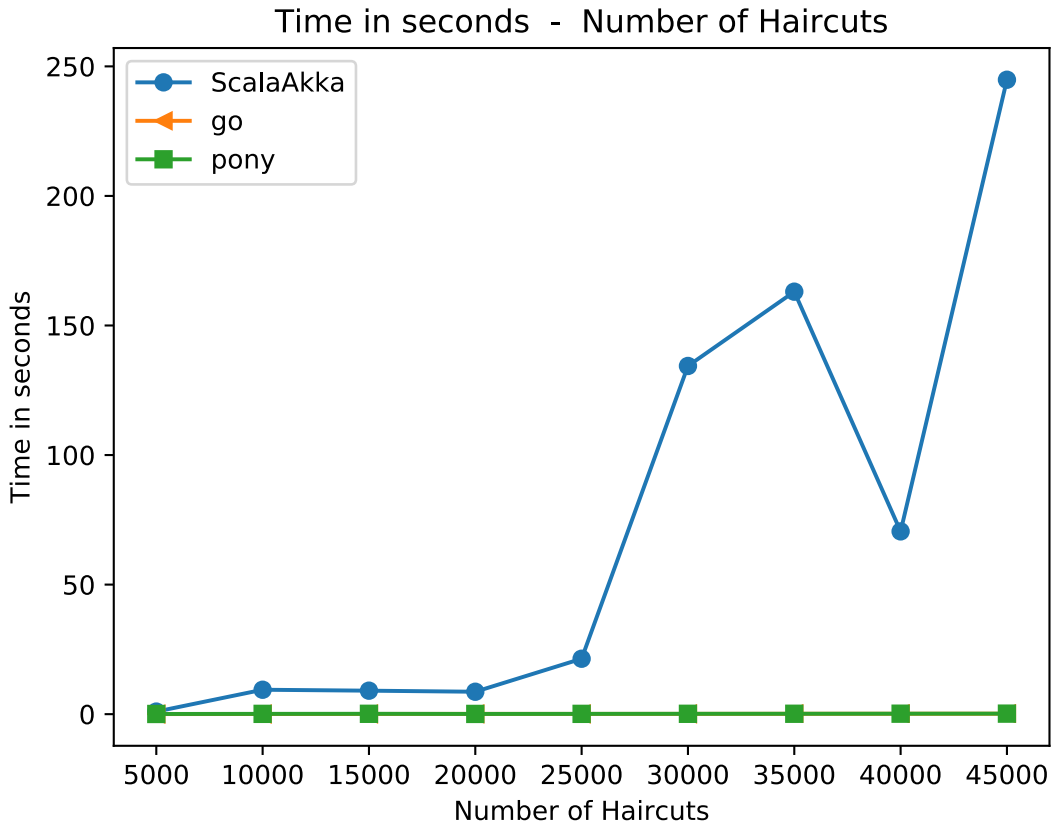
- Ran in a VM, varied number of CPU threads



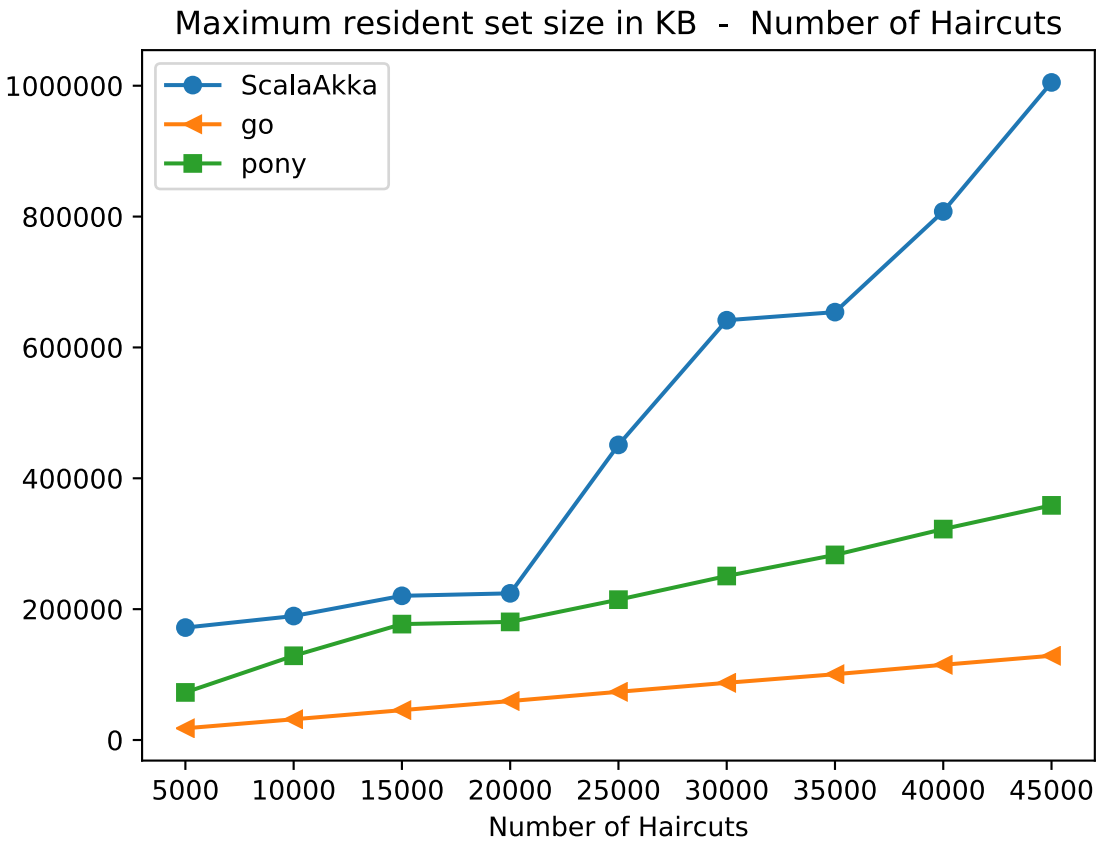




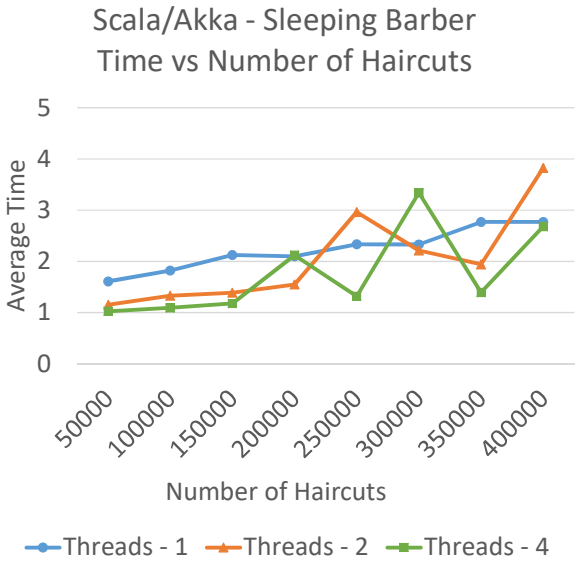
Sleeping Barber Results: Runtime

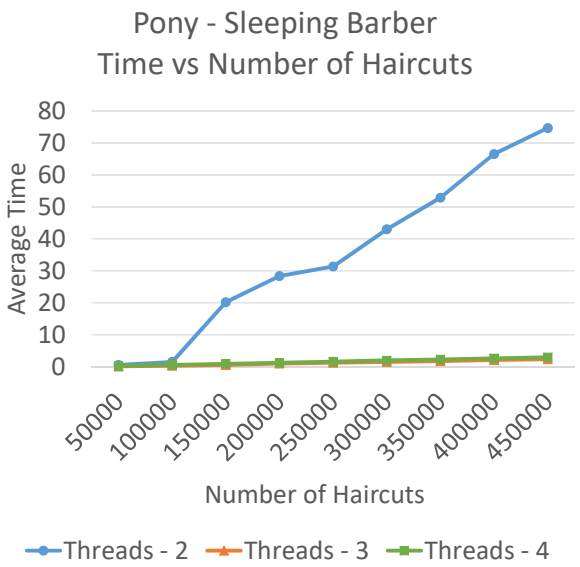


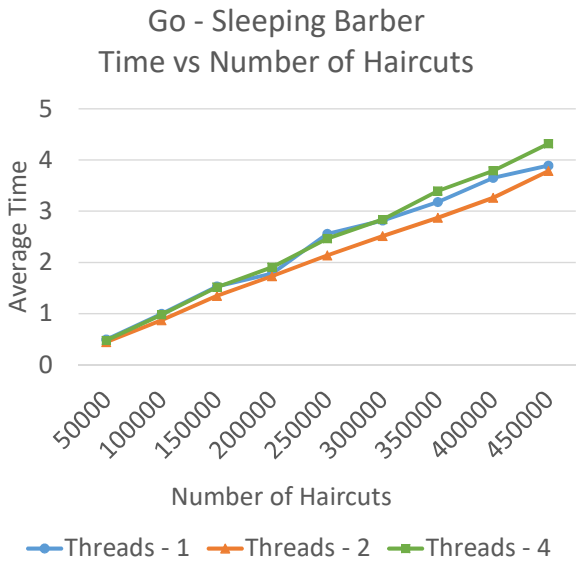
Sleeping Barber Results: Peak Memory Usage



Sleeping Barber Results: CPU Thread Scaling







Machine Specs

Table 3: Machine Specifications

Machine ID	Description	Processor	RAM	OS
Lab-8T	DoC Lab computer	Intel i7-4790, 64-bit, 3.6GHz, 4 cores, 8 threads	16 GB	Ubuntu 16.04.2
Lab-VM-nT	DoC Lab computer (VM, n-threads)	Intel i7-4790, 64-bit, 3.6GHz, 4 cores, 8 threads	8 GB	Ubuntu 17.04
DevX-VM-2T	Developer laptop ThinkPad X230 (VM, 2 threads)	Intel i5-3320M, 64-bit, 2.6GHz, 2 cores, 4 threads	4 GB	Ubuntu 16.04.1
DevT-4T	Developer laptop ThinkPad T430	Intel i5-3320M, 64-bit, 2.6GHz, 2 cores, 4 threads	8 GB	Slackware Linux 14.2
DevMac-4T	Developer laptop MacBook Pro Core i5 2.4 13" Late 2013	Intel i5-4258U, 64-bit, 2.4GHz, 2 cores, 4 threads	4 GB	MacOS Sierra 10.12.4

Results

Table 4: Ping-Pong Test Results

Machine ID	Scala-Akka		Pony		Go	
	n=low	n=high	n=low	n=high	n=low	n=high
Lab-8T	2.30	32.21	0.47	9.42	0.34	7.42
Lab-VM-4T	5.55	93.98	0.44	8.81	0.37	7.58
DevX-VM-2T	3.28	34.01	0.44	9.00	0.48	9.66
DevT-4T	3.07	37.19	0.19	4.91	0.52	10.68
DevMac-4T	5.26	83.85	0.35	6.92	0.67	11.32

variable n = no. of messages (low: 1,000,000, high: 20,000,000)

Table 5: Sleeping Barber Problem Results

Machine ID	Scala-Akka		Pony		Go	
	n=low	n=high	n=low	n=high	n=low	n=high
Lab-8T	3.57	3.19	0.15	0.42	0.03	0.21
Lab-VM-4T	2.03	5.19	0.10	0.27	0.03	0.22
DevX-VM-2T	2.58	23.77	13.79	11.54	0.05	0.35
DevT-4T	1.84	3.63	0.07	0.59	0.03	0.27
DevMac-4T	2.53	15.85	0.15	2.14	0.05	0.39

variable n = no. of haircuts (low: 5,000, high: 45,000)

Table 6: N-Body Simulation (changing no. of bodies) Results

Machine ID	Scala-Akka		Pony		Go	
	n=low	n=high	n=low	n=high	n=low	n=high
Lab-8T	3.47	564.20	1.86	49.47	0.80	52.26
Lab-VM-4T	5.91	798.21	2.14	60.84	0.83	130.65
DevX-VM-2T	10.26	1584.28	4.28	407.62	1.24	199.56
DevT-4T	8.70	1076.69	2.68	98.33	1.15	109.34
DevMac-4T	7.62	1129.61	2.78	106.01	0.06	116.56

fixed no. of steps = 100,000
variable n = no. of bodies (low: 5, high: 95)

Table 7: N-Body Simulation (changing no. of steps) Results

Machine ID	Scala-Akka		Pony		Go	
	n=low	n=high	n=low	n=high	n=low	n=high
Lab-8T	18.70	167.00	2.69	21.71	2.79	22.59
Lab-VM-4T	25.99	232.35	3.11	26.13	4.65	40.66
DevX-VM-2T	54.84	462.01	16.68	146.86	7.19	65.12
DevT-4T	35.26	319.33	4.40	37.46	4.74	40.51
DevMac-4T	38.21	351.94	4.65	39.49	4.08	35.05

fixed no. of bodies = 50
variable n = no. of steps (low: 10,000, high: 100,000)