Universität Heidelberg
Wintersemester 18/19

Group Parallel and Distributed Systems (PVS)
Artur Andrzejak, Diego Costa

Problem Set 2 for lecture Distributed Systems I (IVS1)

Due: 06.11.2018, 14:00 Uhr

Exercise 1 (2 Points)

In a conventional function call, parameters may be passed using copy-by-value, copy-by-reference and even call-by-copy/restore strategy[1]. Consider the following code `newlist = append(data, oldlist)`, where `data` is a numeric value and both `newlist` and `oldlist` are objects from the list/array type. Using this example, describe what mechanisms Python, Java and C use to pass its parameters to the function append. Does the parameter type has any influence on the strategy adopted?

Exercise 2 (1 Point)

Given two processes A and B running in the same machine, why can't process A use pointers or references to pass a parameter in a remote procedule call to process B?

Exercise 3 (2 Points)

Sun Open Network uses a fixed alignment for primitive values (4-byte boundary), whereas CORBA aligns a primitive value of size n on an n-byte boundary. Discuss the trade-offs in choosing the sizes occupied by primitive values.

The next two exercises will use the code presented in Lecture 3 as the codebase. Read the RPyC documentation[2] and expand on the topics presented in class, in special, read the Part 4 (Callbacks and Symmetry) and Part 5 (Asynchronous Operations) to learn how RPyC enables asynchronous communication.

Exercise 4 (3 Points)

Modify the service `get_primes` to incorporate two other parameters: `batch_size` and a `callback` function. The callback should be used to give short feedbacks to the client every time a batch of primes have been calculated. For instance, for a batch_size = 1000, a message could be printed on the client-side:

"1000 primes calculated in X seconds" "2000 primes calculated in Y seconds"

Submit your code in Moodle as your solution.

[1] https://en.wikipedia.org/wiki/Evaluation_strategy
[2] https://rpyc.readthedocs.io/en/latest/tutorial.html

Exercise 5 (5 Points)

So far, all communications between server and client were done synchronously. In this exercise, we will use asynchronous communication to explore local parallelism.

1. Modify the client code to submit asynchronous requests (using a `async_` wrapper). Use asynchronous requests to parallelize the primes number calculation, by dividing the original prime interval into `N` intervals of equal size.
2. To identify whether the server has finished a request, modify the client to either 1) verify the AsyncResult object states or 2) use callback functions. Please refer to the tutorial or the official documentation on async functions[3].
3. Run your program with `N={1,2,..10}` and print the elapsed execution time for each N. What N value yield the fastest prime calculation in your local machine?

Submit your code together with the program outputs for each N in Moodle and test cases.

Exercise 6 (2 Points)

In the additional slides, we describe three distinct RPC systems: Sun Open Network Computing, Distributed Computing Environment and CORBA. Compare the advantages and disadvantages of each system. In what kind of project would you recommend each RPC framework?

[3] https://rpyc.readthedocs.io/en/latest/docs/async.html#async