Methods for disseminating data across servers

turtleneck-master • 3 Apr 2026

Firstly, let us posit we possess a complement of ten servers, each endowed with a capacity of one hundred. We are tasked with the insertion of eighty files, each exhibiting a magnitude of ten.

The most rudimentary approach would be to randomly locate a server with available capacity and populate it with data, let us examine the results via a Python implementation.

```
import random

arr = []
i = 0

for _ in range(10):
    arr.append(100)

while i < 80:
    c = random.randint(0, 9)
    if arr[c] <= 0:
        continue
    arr[c] -= 10
    i += 1

print(arr)
```

This Python code executes a remarkably straightforward task: It establishes a list, christened ‘arr’, populating it with ten instances of the integer 100. Subsequently, it subtracts 10 from the values at non-zero indices, repeating this operation 80 times. Let us now observe the resultant outcome.

```
[0, 80, 0, 0, 50, 20, 10, 10, 0, 30]
```

Envision each index as a server. Some servers may have zero capacity remaining, while others boast eighty. This approach, though straightforward, is remarkably inefficient. Might a superior method exist? Let us delve into the matter.

```
import random

arr = []
i = 0

for _ in range(10):
    arr.append(100)

while i < 80:
    c = random.randint(0, 4)
    c2 = random.randint(5, 9)
    if arr[c] <= 0 and arr[c2] <= 0:
        continue
    if arr[c] > arr[c2]:
        arr[c] -= 10
    elif arr[c] < arr[c2]:
        arr[c2] -= 10
    else:
        if random.randint(0, 1) == 0:
            arr[c] -= 10
        else:
            arr[c2] -= 10
    i += 1

print(arr)
```

This code undertakes to bifurcate ten servers into two distinct groups, thenceforth randomly selecting a solitary server from each. A comparative analysis ensues, diminishing the value of the server boasting superior available space. In the event of parity in available space, the selection is arbitrated randomly. Might this method alone suffice to evince a more felicitous distribution?

```
[20, 20, 20, 30, 10, 20, 20, 20, 10, 30]
```

The results are truly remarkable. Despite its simplicity, server utilization has become exceptionally stable. This technique, known as ‘the power of two choices,’ demonstrates dramatic effects with minimal implementation. Superior algorithms may exist, yet we perpetually grapple with resource constraints. This experiment underscores the indispensable relevance of algorithms.