*CPE 369 Lab 7 investigation result*

Section 03, Waylin Wang, Myron Zhao

# Implementations:

## Sequential

Sequential implementation reads in the JSON objects one at a time and look for the location field. All locations are then parsed into strings and stored as the key in a standard Java Treemap. The choice of Treemap over standard Hashmap is because we wanted to replicate the sort property of Hadoop. The use of Treemap is O(log N) instead of O(N) for Hashmap.

*Pseudo-code:*

Read in JSON object

Get location field information

Translate location information into string with format: (x, y)

Perform lookup and insert into Treemap

Print out entire Treemap’s key value pair to file.

## Hadoop

Hadoop implementation is rather straight forward. Pseudo-code is provided below.

*Pseudo-code:*

Map:

Read in JSON object

Parse location information into string with format: (x, y)

Emit location information string as key and 1 as value

Reduce:

For every values in the key, sum up the values.

Emit the key (location) as key and sum as value.

# Result

The test was conducted with 1000 to 8 million objects. Each file was ran for five times with each implementation and the average time was used in the analysis.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num Objects | Hadoop 1 | Hadoop 2 | Hadoop 3 | Hadoop 4 | Hadoop 5 | Avg | Std Dev |
| 1000 | 34215 | 31697 | 33634 | 32346 | 34321 | 33242.6 | 1167.740682 |
| 2000 | 37742 | 33602 | 32339 | 32525 | 32665 | 33774.6 | 2270.666928 |
| 4000 | 33841 | 33592 | 34373 | 33371 | 33350 | 33705.4 | 422.808822 |
| 8000 | 34433 | 34859 | 33434 | 34498 | 33584 | 34161.6 | 619.7203402 |
| 16000 | 35467 | 34576 | 35305 | 35545 | 34565 | 35091.6 | 483.5253871 |
| 32000 | 34448 | 34439 | 34342 | 34544 | 35604 | 34675.4 | 524.0026717 |
| 64000 | 36408 | 35457 | 35619 | 35372 | 34730 | 35517.2 | 601.7222781 |
| 128000 | 37504 | 36617 | 37485 | 37458 | 37489 | 37310.6 | 388.0892938 |
| 256000 | 39520 | 39534 | 39620 | 39606 | 38451 | 39346.2 | 502.3247953 |
| 512000 | 44655 | 45579 | 45834 | 44556 | 45640 | 45252.8 | 599.3785949 |
| 1024000 | 56663 | 55657 | 54691 | 54511 | 56028 | 55510 | 906.656495 |
| 2048000 | 54528 | 54716 | 53753 | 55587 | 55509 | 54818.6 | 757.886733 |
| 4096000 | 88929 | 95830 | 92891 | 88734 | 93830 | 92042.8 | 3118.506004 |
| 8192000 | 184471 | 193426 | 183606 | 189300 | 197360 | 189632.6 | 5856.091683 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num Objects | Seq1 | Seq 2 | Seq3 | Seq 4 | Seq 5 | Avg | Std Dev |
| 1000 | 288 | 295 | 324 | 289 | 288 | 296.8 | 15.4822479 |
| 2000 | 323 | 325 | 350 | 343 | 346 | 337.4 | 12.50199984 |
| 4000 | 372 | 381 | 409 | 391 | 398 | 390.2 | 14.41180072 |
| 8000 | 435 | 437 | 411 | 414 | 421 | 423.6 | 11.90798052 |
| 16000 | 528 | 539 | 534 | 537 | 529 | 533.4 | 4.827007354 |
| 32000 | 640 | 642 | 654 | 626 | 631 | 638.6 | 10.80740487 |
| 64000 | 905 | 925 | 942 | 847 | 885 | 900.8 | 36.89444403 |
| 128000 | 1286 | 1347 | 1292 | 1261 | 1287 | 1294.6 | 31.67491121 |
| 256000 | 2030 | 2087 | 2092 | 2050 | 2058 | 2063.4 | 25.97691283 |
| 512000 | 3626 | 3680 | 3712 | 3767 | 3692 | 3695.4 | 51.15466743 |
| 1024000 | 6575 | 6775 | 6557 | 6628 | 6710 | 6649 | 92.16561181 |
| 2048000 | 14041 | 13812 | 13573 | 13730 | 13730 | 13777.2 | 170.9932747 |
| 4096000 | 27145 | 26927 | 27599 | 26459 | 26940 | 27014 | 412.4487847 |
| 8192000 | 57957 | 52353 | 52241 | 55479 | 53379 | 54281.8 | 2431.078197 |

# Analysis

The sequential implementation beat Hadoop’s implementation for every test conducted. Through regression analysis, the sequential implementation is following a linear growth time, with R being .9999. Hadoop implementation is following almost a linear growth time with R = .97552. A little bit of online research did show that unless we are working with multiple terabytes of data, we will see no benefits of using Hadoop rather than a straightforward implementation.