

Performance Testing

Aan het begin van dit project was er een analyse opgesteld over het implementeren van een Crawler Parser vs een gecombineerde Crawler Parser. Ofwel, twee verschillende processen tegenover een proces. Deze analyse is te vinden in het Hap Plan.

Om de performance van deze twee verschillende processen te testen hebben wij op de volgende punten gelet:

Onderdelen	Beschrijving
CPU tijd	Hoeveel CPU tijd is heeft master proces nodig om de queues bij te houden tijdens het verloop van de test
Verwerkte data	Kijken hoeveel elementen er in de queues zijn of zijn geweest. Hoeveel data is er verwerkt.

Performance test case

Om de performance te testen hebben de volgende test ontworpen:

Stap	Uitleg
Start het master proces op 1 PI	
Start de Crawler & Parser of CrawlerParser processen op één aparte PI	Hiervoor hebben wij gekozen voor: 3x Crawler & 3x Parser. Al deze processen worden gelijktijdig op de zelfde PI uitgevoerd. Bij de combined test waren dit 3 processen, ook op de zelfde PI.
Laat de test 15 minuten lopen	Tijdens deze 15 minuten worden er logs gegenereerd. Ook wordt de data verwerkt. Achteraf wordt dit gebruikt voor het vergelijken van de data.
Haal uit de database op hoeveel data er is verwerkt	Dit zijn de row counts van de volgende tabellen: <ul style="list-style-type: none">• Indices• Uri_queue• Uris• Document_queue

Uitvoering

Als eerste hebben wij de performance test uitgevoerd met het CombinedCrawlerParser proces en vervolgens de verschillende processen (Aparte Crawler & Parser). Hier onder zijn de resultaten van deze testen te vinden weer gegeven in tabellen.

Verwerkte data

Tabel	CombinedCrawlerParser	Aparte Crawler & Parser
Indices	33115	9100
Uri_queue	5493	758
Uris	5673	824

CPU Tijd

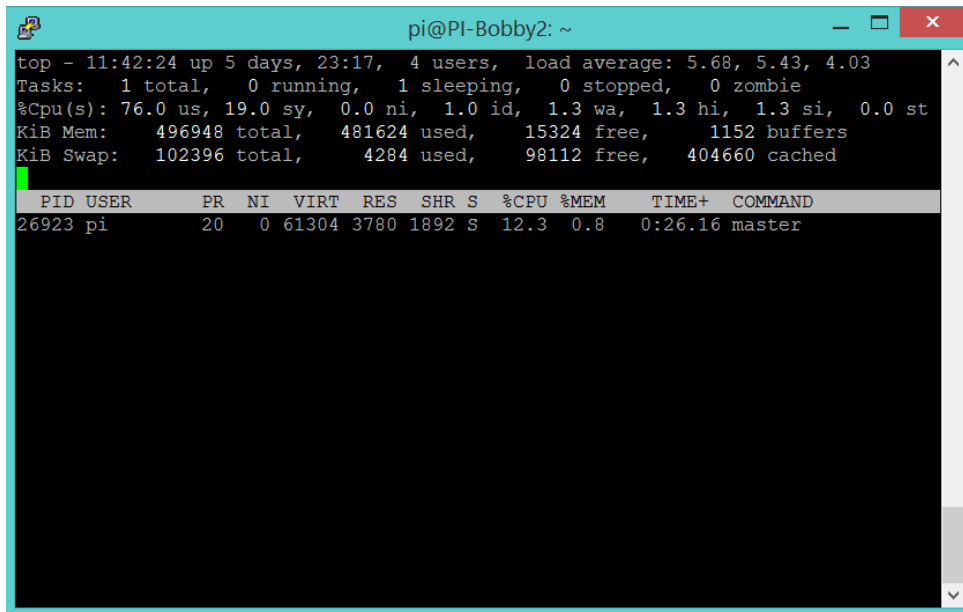
CombinedCrawlerParser	Aparte Crawler & Parser
0:26.16	2:15.11

Resultaat

Als conclusie uit de performance tests komt dat een gecombineerde Crawler & Parser proces sneller data verwerkt dan twee verschillende processen, binnen de zelfde tijd periode en vanaf het zelfde start punt.

Een van de mogelijke redenen waarom het gecombineerde proces sneller is, is mogelijk omdat hierdoor de documenten queue niet meer wordt gebruikt. Alle data wordt gelijk verwerkt en niet eerst opgeslagen bij de master. Dit zorgt ook voor minder I/O en gebruik van de database.

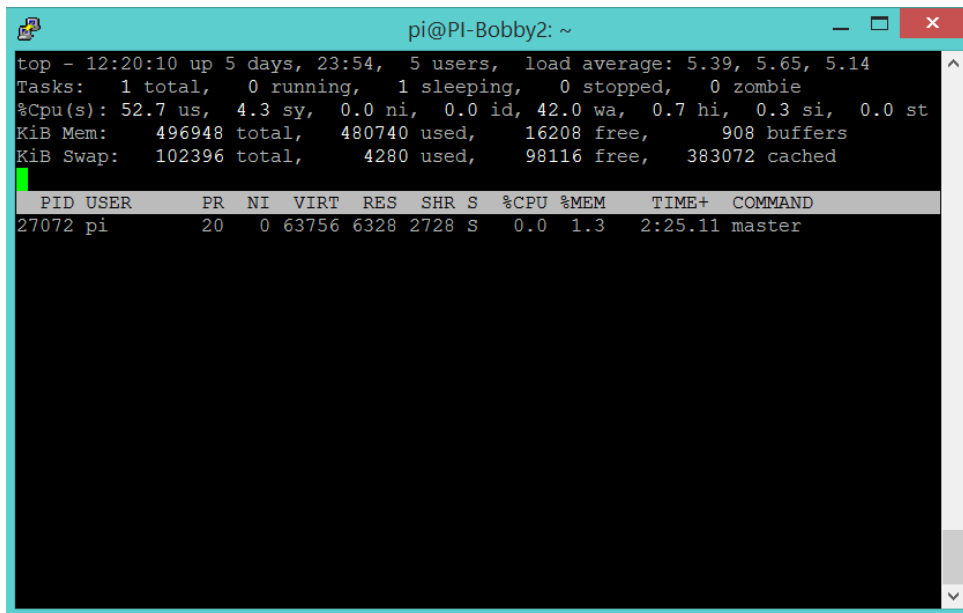
Bijlage 1 - screenshots



```
pi@PI-Bobby2: ~  
top - 11:42:24 up 5 days, 23:17,  4 users,  load average: 5.68, 5.43, 4.03  
Tasks:  1 total,   0 running,   1 sleeping,   0 stopped,   0 zombie  
%Cpu(s): 76.0 us, 19.0 sy,  0.0 ni,  1.0 id,  1.3 wa,  1.3 hi,  1.3 si,  0.0 st  
KiB Mem:  496948 total,  481624 used,   15324 free,   1152 buffers  
KiB Swap:  102396 total,    4284 used,   98112 free,  404660 cached  


| PID   | USER | PR | NI | VIRT  | RES  | SHR  | S | %CPU | %MEM | TIME+   | COMMAND |
|-------|------|----|----|-------|------|------|---|------|------|---------|---------|
| 26923 | pi   | 20 | 0  | 61304 | 3780 | 1892 | S | 12.3 | 0.8  | 0:26.16 | master  |


```



```
pi@PI-Bobby2: ~  
top - 12:20:10 up 5 days, 23:54,  5 users,  load average: 5.39, 5.65, 5.14  
Tasks:  1 total,   0 running,   1 sleeping,   0 stopped,   0 zombie  
%Cpu(s): 52.7 us,  4.3 sy,  0.0 ni,  0.0 id, 42.0 wa,  0.7 hi,  0.3 si,  0.0 st  
KiB Mem:  496948 total,  480740 used,   16208 free,    908 buffers  
KiB Swap:  102396 total,    4280 used,   98116 free,  383072 cached  


| PID   | USER | PR | NI | VIRT  | RES  | SHR  | S | %CPU | %MEM | TIME+   | COMMAND |
|-------|------|----|----|-------|------|------|---|------|------|---------|---------|
| 27072 | pi   | 20 | 0  | 63756 | 6328 | 2728 | S | 0.0  | 1.3  | 2:25.11 | master  |


```