进程和线程实验报告

物理试验班 51 李昂 2150201074

实验原理

通过编写多进程的并发服务器-客户端程序和多线程的并发服务器-客户端程序来比较服务器端资源(主要是内存)的开销,进而体现线程和进程在资源使用上的不同。

实验平台: Macos Catalina 10.15

代码实现

编写服务器端和客户端程序,运用socket网络编程实现本机端口间的通信。当连接建立后 ,服务器分别创建新的进程和线程来处理客户端的请求,此处为简单的发送日期和时间。 代码见附件

客户端 tcp client.cpp

服务器端(多进程并发) tcp_server_process.cpp

服务器端(多线程并发) tcp_server_thread.cpp

实验过程

执行编译

clang++ -o tcp_server_process tcp_server_process.cpp clang++ -o tcp_server_thread -lpthread tcp_server_thread.cpp //由于使用了pthread库所以编译时要加上-lpthread.

clang++ -o tcp client tcp client.cpp

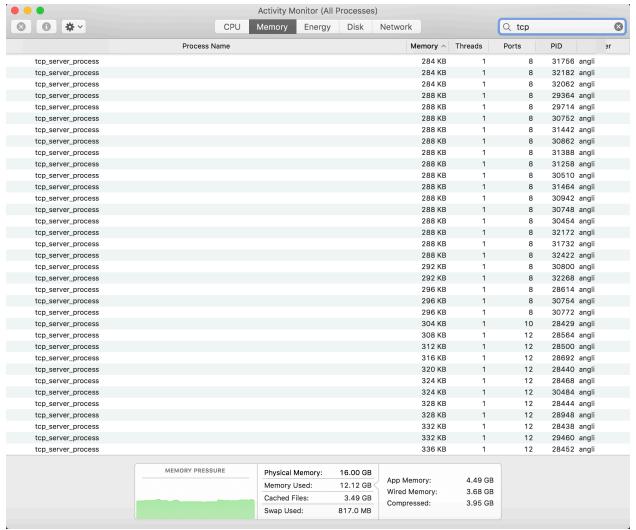
1.使用多进程并发服务器端

./tcp_server_process 12321 #服务器端监听本地127.0.0.1的12321端口 ./tcp_client 127.0.0.1 12321

#客户端向本地127.0.0.1的12321端口发出请求,通过运行client.sh 让客户端请求2000次

37%	_ II 8.9 GB ½ 2.1 kB↓	1.0 k
< tcp_server_proce	≡ × bash	
lient:127.0.0.1(63286)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32392	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63287)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32394	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63288)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32396	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63289)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32398	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63290)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32400	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63291)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32402	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63292)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32404	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63293)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32406	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63294)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32408	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63295)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32410	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63296)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32412	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63297)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32414	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63298)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32416	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63299)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32418	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63300)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32420	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63301)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32422	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63302)connected	Wed Oct 9 20:30:59 2019	
'm a new process and my pid is 32424	Wed Oct 9 20:30:59 2019	
lient:127.0.0.1(63303)connected	Wed Oct 9 20:31:00 2019	
'm a new process and my pid is 32426	Wed Oct 9 20:31:00 2019	
lient:127.0.0.1(63304)connected	Wed Oct 9 20:31:00 2019	
'm a new process and my pid is 32428	Wed Oct 9 20:31:00 2019	
lient:127.0.0.1(63305)connected	Wed Oct 9 20:31:00 2019	
'm a new process and my pid is 32430	Wed Oct 9 20:31:00 2019	
lient:127.0.0.1(63306)connected	Wed Oct 9 20:31:00 2019	
'm a new process and my pid is 32432	Wed Oct 9 20:31:00 2019	
lient:127.0.0.1(63307)connected	Wed Oct 9 20:31:00 2019	
'm a new process and my pid is 32434	Wed Oct 9 20:31:00 2019	
lient:127.0.0.1(63308)connected	Wed Oct 9 20:31:00 2019	
'm a new process and my pid is 32436	Wed Oct 9 20:31:00 2019	
lient:127.0.0.1(63309)connected	Wed Oct 9 20:31:00 2019	
	►(base) Angs-MacBook-Pro:processysthread angli\$	

左边为服务器端,右边为客户端。由左图可知对每一次客户端的请求服务器端都创建了一个新进程来打印客户端信息、自己的进程pid和向客户端发送时间日期。现在我们来看内存占用:

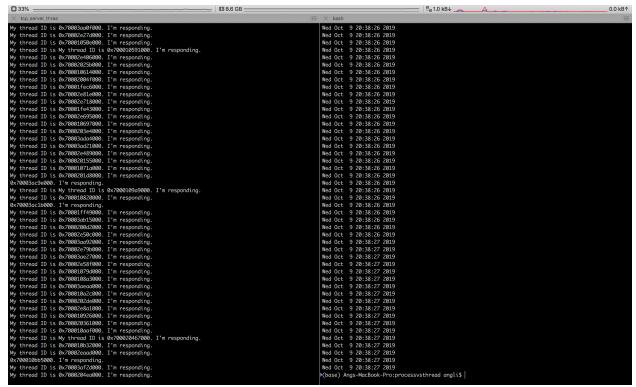


每个进程的线程数为1,每个进程所占用的内存资源在240KB到340KB之间,平均为每个进程约为300KB,而这样的进程有2000个,共占用约600MB内存资源。

2.使用多线程并发服务器端

./tcp_server_thread 12321 #服务器端监听本地127.0.0.1的12321端口 ./tcp_client 127.0.0.1 12321

#客户端向本地127.0.0.1的12321端口发出请求,通过运行client.sh 让客户端请求2000次



左边为服务器端,右边为客户端。由左图可知对每一次客户端的请求服务器端都创建了一个新线程来打印客户端信息(此处已经被后面的循环覆盖了)、自己的线程id和向客户端发送时间日期。

现在我们来看资源占用



这次就只有一个进程,这一个进程中有2001个线程,总内存占用为16.1MB。

实验结论

通过这两组实验对比,同样是客户端发出2000次请求,多进程并发客户端消耗了约600M B的内存资源,而多线程并发客户端只消耗了16.1MB的内存资源。可见通过多线程并发能 显著降低对内存资源的消耗。

参考资料:

https://www.cnblogs.com/SeekHit/p/6537932.html Stack Overflow www.geeksforgeeks.org CSDN 《Operating.System.Concepts.10th.Edition》

本实验 GitHub:

https://github.com/ang830715/processvsthread.git