Network Programming for Windows 06:

Scalable Winsock Applications

jintaeks@dongseo.ac.kr

Division of Digital Contents, DSU

August 2017

DSU Dongseo Univ 동서대학교

Scalable Server Architectures

- ✓ We'll get into the details of implementing a scalable server.
- ✓ Focuses on connection-oriented protocols such as TCP/IP.

Accepting Connections

- ✓ The Microsoft extension **AcceptEx**() is the only Winsock function capable of accepting a client connection via overlapped I/O.
- ✓ A responsive server must always have **enough AcceptEx() calls outstanding** so that incoming client connections may be handled immediately.
- ✓ **For Windows NT Server, the maximum backlog value is currently 200.**
- ✓ When creating the **listening socket**, associate it with an event by using the **WSAEventSelect**() API call and registering for FD_ACCEPT notification.
  - If there are no pending AcceptEx() operations but there are incoming client connections (accepted by the system according to the backlog value), then the event will be signaled

- ✓ Although it may seem logical and simpler to post **AcceptEx**() requests in **one of the worker threads** handling notification from the completion port, you should avoid this because socket creation process is expensive.
- ✓ In addition, any **complex computations should be avoided within the worker threads** so the server may process the completion notifications as fast as possible.

Data Transfers

- ✓ All data sent or received should be performed with overlapped I/O.
- ✓ Each socket has an associated **send and receive buffer** that is used to buffer outgoing and incoming data, respectively.
  - WSA_IO_PENDING

TransmitFile() and TransmitPackets()

- ✓ The benefit of these functions is that a **great deal of data** can be queued for sending on a connection while incurring just a **single user-to-kernel mode transition**.

Resource Management

- ✓ The server handles increasingly more concurrent connections, a **resource limitation** will eventually be encountered.
- ✓ The two limits most likely to be encountered are **the number of locked pages** and **non-paged pool** usage.

number of locked-pages

- ✓ With every overlapped send or receive operation, it is probable that the data buffers submitted will be **locked**.
  - – It cannot be paged out of physical memory.
- ✓ Another important consideration is the **page size** on the architecture the server is running on.
- ✓ When the system **locks** memory passed into overlapped operations, it does so on page boundaries.
- ✓ On the **x86** architecture, pages are locked in **multiples of 4 KB**.
- ✓ If an operation posts a 1 KB buffer, then the system is actually locking a 4 KB chunk of memory.

non-paged pool

- ✓ Hitting the **non-paged pool limit** is a much more serious error and is difficult to recover from.
- ✓ Non-paged pool is the portion of memory that is always resident in physical memory and **can never be paged out**.
- ✓ Each socket created consumes a small portion of non-paged pool that is used to maintain socket state information.
  - – A connected socket: about 2 KB of non-paged pool.
  - – A socket returned from accept or AcceptEx(): 1.5 KB.
  - – Each overlapped operation issued on a socket: 500 bytes.
- ✓ If the system does **run out of non-paged pool**.
  - – In the best-case scenario, Winsock calls will fail with WSAENOBUFS.
  - – The worst-case scenario is the system crashes with a terminal error.

Server Stragegies

- ✓ Servers can be divided roughly into two categories:
  - **High throughput**
  - **High connections**
- ✓ A high throughput server
  - More concerned with pushing data on a small number of connections.
  - An **FTP** server is an example of a high throughput server.
- ✓ A high connection server
  - More concerned with handling a large number of connections and is not attempting to push large data amounts.
  - An example of this would be an **instant messenger server**.

Performance Numbers

I/O Model	Attempted/Connected	Memory Used (KB)	Non-Paged Pool	CPU Usage	Threads	Throughput (Send/ Receive Bytes Per Second)
Blocking	7000/ 1008	25,632	36,121	10-60%	2016	2,198,148/ 2,198,148
	12,000/ 1008	25,408	36,352	5- 40%	2016	404,227/ 402,227
Non-blocking	7000/ 4011	4208	135,123	95-100%*	1	0/0
	12,000/ 5779	5224	156,260	95-100%*	1	0/0
WSAAsync Select	7000/ 1956	3640	38,246	75-85%	3	1,610,204/ 1,637,819
	12,000/ 4077	4884	42,992	90-100%	3	652,902/ 652,902
WSAEvent Select	7000/ 6999	10,502	36,402	65-85%	113	4,921,350/ 5,186,297
	12,000/ 11,080	19,214	39,040	50-60%	192	3,217,493/ 3,217,493
	46,000/ 45,933	37,392	121,624	80-90%	791	3,851,059/ 3,851,059
Overlapped (events)	7000/ 5558	21,844	34,944	65-85%	66	5,024,723/ 4,095,644
	12,000/12,000	60,576	48,060	35-45%	195	1,803,878/ 1,803,878
	49,000/48,997	241,208	155,480	85-95%	792	3,865,152/ 3,834,511
Overlapped (completion port)	7000/ 7000	36,160	31,128	40-50%	2	6,282,473/ 3,893,507
	12,000/12,000	59,256	38,862	40-50%	2	5,027,914/ 5,027,095
	50,000/49,997	242,272	148,192	55-65%	2	4,326,946/ 4,326,496

Practice

- ✓ chapter06 → iocpServer project

MY **BRIGHT** FUTURE

DSU Dongseo University
동서대학교