IPC performance: Named Pipe vs Socket

Everyone seems to say named pipes are faster than sockets IPC. How much faster are they? I would prefer to use sockets because they can do two-way communication and are very flexible but will choose speed over flexibility if it is by considerable amount.

linux performance sockets ipc named-pipes

edited Jun 30 '12 at 3:59
Tshepang
4,905 ● 11 ● 60 ● 104

asked Aug 5 '09 at 21:52
user19745
1,135 ● 4 ● 16 ● 20

8 Your mileage will vary. :) Profile typical use for your intended application, and pick the better of the two. Then profile anonymous pipes, sockets of other domains and families, semaphores and shared memory or message queues (SysV and POSIX), realtime signals with a word of data, or whatever. `pipe(2)` (or, `mkfifo(3)`?) may be the winner, but you won't know until you try. – pilcrow Aug 6 '09 at 1:21

1 SysV message queues FTW! I have no idea if they're fast, i just have a soft spot for them. – Tom Anderson Sep 21 '10 at 17:44

1 What is "speed" in this case? Overall data transfer rate? Or latency (how quickly the first byte gets to the receiver)? If you want fast local data transfer, then it's hard to beat shared memory. If latency is an issue, though, then the question gets more interesting... – Ian Ni-Lewis Mar 30 '16 at 20:28

8 Answers

I would suggest you take the easy path first, carefully isolating the ipc mechanism so that you can change from socket to pipe, but I would definitely go with socket first. You should be sure IPC performance is a problem before preemptively optimizing.

And if you get in trouble because of IPC speed, I think you should consider switching to shared memory rather rather than going to pipe.

If you want to do some transfer speed testing, you should try socat, which is a very versatile program that allows you to create almost any kind of tunnel.

answered Aug 6 '09 at 13:17
shodanex
9,154 ● 8 ● 38 ● 70

+1 for socat, hadn't seen that before – galaktor Jan 15 '14 at 8:43

I'm going to agree with shodanex, it looks like you're prematurely trying to optimize something that isn't yet problematic. Unless you *know* sockets are going to be a bottleneck, I'd just use them.

A lot of people who swear by named pipes find a little savings (depending on how well everything else is written), but end up with code that spends more time blocking for an IPC reply than it does doing useful work. Sure, non-blocking schemes help this, but those can be tricky. Spending years bringing old code into the modern age, I can say, the speedup is almost nil in the majority of cases I've seen.

If you really think that sockets are going to slow you down, then go out of the gate using shared memory with careful attention to how you use locks. Again, in all actuality, you might find a small speedup, but notice that you're wasting a portion of it waiting on mutual exclusion locks. I'm not going to advocate a trip to futex hell (well, not *quite* hell anymore in 2015, depending upon your experience).

Pound for pound, sockets are (almost) always the best way to go for user space IPC under a monolithic kernel .. and (usually) the easiest to debug and maintain.

edited May 31 '16 at 18:30

answered Aug 7 '09 at 17:52
Tim Post ♦
25.2k ● 12 ● 84 ● 149

Keep in mind that sockets does not necessarily mean IP (and TCP or UDP): You can also use UNIX sockets (PF_UNIX), which offer a noticeable performance improvement over connecting to 127.0.0.1

answered Jul 22 '11 at 2:34
Yuliy
11.6k ● 3 ● 29 ● 42

As often, numbers says more than feeling, here are some data: Pipe vs Unix Socket Performance (opendmx.net).

This benchmark shows a difference of about 12 to 15% faster speed for pipes.

edited Jul 1 '13 at 11:59
slashmais
4,436 ● 7 ● 39 ● 61

answered Mar 24 '13 at 20:24
Hibou57
1,849 ● 1 ● 24 ● 31

If you do not need speed, sockets are the easiest way to go!

If what you are looking at is speed, the fastest solution is shared Memory, not named pipes.

answered Dec 23 '09 at 20:55
Damien
654 ● 7 ● 17

For two way communication with named pipes:

- If you have few processes, you can open two pipes for two directions (processA2ProcessB and processB2ProcessA)
- If you have many processes, you can open in and out pipes for every process (processAin, processAout, processBin, processBout, processCin, processCout etc)
- Or you can go hybrid as always :)

Named pipes are quite easy to implement.

E.g. I implemented a project in C with named pipes, thanks to standart file input-output based communication (fopen, fprintf, fscanf ...) it was so easy and clean (if that is also a consideration).

I even coded them with java (I was serializing and sending objects over them!)

Named pipes has one disadvantage:

- they do not scale on multiple computers like sockets since they rely on filesystem (assuming shared filesystem is not an option)

answered Sep 21 '10 at 17:28
daghan
626 ● 6 ● 14

Named pipes and sockets are not functionally equivalent; sockets provide more features (they are bidirectional, for a start).

We cannot tell you which will perform better, but I strongly suspect it doesn't matter.

Unix domain sockets will do pretty much what tcp sockets will, but only on the local machine and with (perhaps a bit) lower overhead.

If a Unix socket isn't fast enough and you're transferring a lot of data, consider using shared memory between your client and server (which is a LOT more complicated to set up).

Unix and NT both have "Named pipes" but they are totally different in feature set.

answered Aug 6 '09 at 7:26
MarkR
47.1k ● 8 ● 88 ● 121

One problem with pipes is that they do not have a way to flush the buffer. There is something called the Nagle algorithm which collects all data and flushes it after 40ms. So if it is responsiveness and not bandwidth you might be better off with a pipe.

You can disable the Nagle with the socket option TCP_NODELAY but then the reading end will never receive two short messages in one single read call.

So test it, i ended up with none of this and implemented memory mapped based queues with pthread mutex and semaphore in shared memory, avoiding a lot of kernel system calls (but today they aren't very slow anymore).

edited Apr 18 '16 at 9:04

answered Jun 20 '15 at 15:44
Lothar
5,738 ● 4 ● 42 ● 77

"So test it" <-- words to live by. – Koshinae Apr 18 '16 at 9:03