What is wrong with polling?

I have heard a few developers recently say that they are simply polling stuff (databases, files, etc.) to determine when something has changed and then run a task, such as an import.

I'm really against this idea and feel that utilising available technology such as Remoting, WCF, etc. would be far better than polling.

However, I'd like to identify the reasons why other people prefer one approach over the other and more importantly, how can I convince others that polling is wrong in this day and age?

c# .net asp.net polling

edited Nov 10 '10 at 20:09
Peter Mortensen
7,802 8 55 90

asked Nov 26 '08 at 10:52
HAdes
4,601 10 37 63

FYI: Remoting and WCF do polling. – Timothy Khouri Nov 26 '08 at 12:27

To a certain degree yes, but not in the same manner in which some developers explicity use polling i.e. polling a database every minute. – HAdes Nov 26 '08 at 23:18

I have similar situation where am polling multiple times on multiple ftp locations to get updated file, what would be an optimal way to deal with the situation? – Rachel Feb 5 '12 at 20:19

15 Answers

Polling is not "wrong" as such.

A lot depends on how it is implemented and for what purpose. If you really care about immedatly notification of a change, it is very efficient. Your code sits in tight loop, constantly polling (asking) a resource whether it has changed / updated. This means you are notified as soon as you can be that something is different. But, your code is not doing anything else and there is overhead in terms of many many calls to the object in question.

If you are less concerned with immediate notification you can increase the interval between polls, and this can also work well, but picking the correct interval can be difficult. Too long and you might miss critical changes, too short and you are back to the problems of the first method.

Alternatives, such as interrupts or messages, etc. can provide a better compromise in these situations. You are notified of a change as soon as is practically possible, but this delay is not something you control, it depends on the component tself being timely about passing on changes in state.

What is "wrong" with polling?

- It can be resource hogging.
- It can be limiting (especially if you have many things you want to know about / poll).
- It can be overkill.

But...

- It is not inherently wrong.
- It can be very effective.
- It is very simple.

edited Nov 10 '10 at 20:11 answered Nov 26 '08 at 10:59


Peter Mortensen
7,802 8 55 90


xan
5,217 4 27 51

1 "This means you are notified as soon as you can be that something is different." Wouldn't that kind of depend on the polling interval and how close you happen to be to when whatever happens? – Svish Aug 27 '09 at 8:52

3 There are certain situations where polling can be more efficient than messages. It depends on number and frequency of "events". As a somewhat contrived example, consider a thermometer that measures some temperature 100 times a second and send an update. If you use separate message for each update, you need to handle 100 messages per second. But suppose you only care to measure the temperature once every 5 seconds. In such a case polling once each 5 seconds will be more efficient than handling 500 messages. – Gregory Mostizky Aug 27 '09 at 8:53

There are two reasons why polling could be considered bad by principle.

1. It is a waste of resources. It is very likely that you will check for a change while no change has occurred. The CPU cycles/bandwidth spend on this action does not result in a change and thus could have been better spend on something else.
2. Polling is done on a certain interval. This means that you won't know that a change has occurred until the next time that the interval has passed.

It would be better to be notified of changes. This way you're not polling for changes that haven't occurred and you'll know of a change as soon as you receive the notification.

answered Nov 26 '08 at 11:02


hopla
738 1 7 12

1 Note that sometimes polling saves resources. It sets an *upper* limit on how often you do the heavy lifting, namely the polling interval. Not often relevant, but when it is relevant it might be important. – Steve Jessop Nov 26 '08 at 11:12

6 Of course, most things that provide a notification of change are themselves polling in order to detect that change. At some point, in many situations, the polling comes first. – Jeff Yates Nov 26 '08 at 13:57

Examples of things that use polling in this day and age:

- Email clients poll for new messages (even with IMAP).
- RSS readers poll for changes to feeds.
- Search engines poll for changes to the pages they index.
- StackOverflow users poll for new questions, by hitting 'refresh' ;-)
- Bittorrent clients poll the tracker (and each other, I think, with DHT) for changes in the swarm.
- Spinlocks on multi-core systems can be the most efficient synchronisation between cores, in cases where the delay is too short for there to be time to schedule another thread on this core, before the other core does whatever we're waiting for.

Sometimes there simply isn't any way to get asynchronous notifications: for example to replace RSS with a push system, the server would have to know about everyone who reads the feed and have a way of contacting them. This is a mailing list - precisely one of the things RSS was designed to avoid. Hence the fact that most of my examples are network apps, where this is most likely to be an issue.

Other times, polling is cheap enough to work even where there is async notification.

For a local file, notification of changes is likely to be the better option in principle. For example, you might (might) prevent the disk spinning down if you're forever poking it, although then again the OS might cache. And if you're polling every second on a file which only changes once an hour, you might be needlessly occupying 0.001% (or whatever) of your machine's processing power. This sounds tiny, but what happens when there are 100,000

files you need to poll?

In practice, though, the overhead is likely to be negligible whichever you do, making it hard to get excited about changing code that currently works. Best thing is to watch out for specific problems that polling causes on the system you want to change - if you find any then raise those rather than trying to make a general argument against all polling. If you don't find any, then you can't fix what isn't broken...

edited Sep 12 '11 at 8:26

answered Nov 26 '08 at 11:32

Steve Jessop
175k 15 235 488

1 OMG!! I am a polling device. :O – EMBarbosa Dec 27 '11 at 17:08

Polling is easy to do, very easy, its as easy as any procedural code. Not polling means you enter the world of Asynchronous programming, which isn't as brain-dead easy, and might even become challenging at times.

And as with everything in any system, the path of less resistance is normally more commonly taken, so there will always be programmers using polling, even great programmers, because sometimes there is no need to complicate things with asynchronous patterns.

I for one always thrive to avoid polling, but sometimes I do polling anyways, especially when the actual gains of asynchronous handling aren't that great, such as when acting against some small local data (of course you get a bit faster, but users won't notice the difference in a case like this). So there is room for both methodologies IMHO.

answered Nov 26 '08 at 11:00

Robert Gould
31.1k 36 131 236

2 +1 for common sense: "sometimes there is no need to complicate things with asynchronous patterns." – JeffK Nov 26 '08 at 18:58

Client polling doesn't scale as well as server notifications. Imagine thousands of clients asking the server "any new data?" every 5 seconds. Now imagine the server keeping a list of clients to notify of new data. Server notification scales better.

answered Apr 9 '09 at 17:39

Slidell4life
172 2 4

Excelent point there, I find in modern times that programmers are sloppy with processor, space, and bandwidth issues, because they have this funny idea that those things will infinitely improve, but eventually we will reach the point where it is impossible to improve those things anymore. Asynchronous notification isn't the devil people make it out to be, anyone programming a GUI on a modern OS deals with asynchronous notifications all the time. – user109878 Jun 2 '09 at 1:55

I think people should realize that in most cases, at some level there is polling being done, even in event or interrupt driven situations, but you're isolated from the actual code doing the polling. Really, this is the most desirable situation ... isolate yourself from the implementaion, and just deal with the event. Even if you must implement the polling yourself, write the code so that it's isolated, and the results are dealt with independently of the implementation.

edited Nov 26 '08 at 17:42

answered Nov 26 '08 at 12:01

Nerdfest
1,444 9 15

This is not always accurate. In the case where an objects properties change, the object itself can setup a notification that occurs when the property is set. In that case polling wouldn't necessarily occur, though it certainly could in the case the object actually checks for a difference between old and new values. – user109878 Jun 2 '09 at 1:38

Its simple - polling is bad - inefficient, waste of resources, etc. There is always some form of connectivity in place that is monitoring for an event of some sort anyway, even if 'polling' is not chosen.

So why go the extra mile and put additional polling in place.

Callbacks are the best option - just need to worry about tie the callback in with your current process. Underlying, there is polling going on to see that the connection is still in place anyhow.

If you keep phoning/ringing your girlfriend and shes never answers, then why keep calling? Just leave a message, and wait until she 'calls back' ;)

answered Nov 26 '08 at 11:12
DJAdes

But what if there's an emergency and you HAVE to find out something from you girlfriend asap? What if she doesn't check her answerphone very often? In this situation you would be relying on her to get back to you, and you have no control over that. I take your point, just giving a counter-example. – xan Nov 26 '08 at 11:21

You could put controls in place to ensure the remote object is always in a valid state, or report problems that cant be sorted itself. Or continuing the analogy, have the answerphone continously beep so the gfriend notices it straight away, or have it send a txt to warn others she may be in trouble. – HAdes Nov 26 '08 at 23:38

Great analogy there. – user109878 Jun 2 '09 at 2:00

I use polling occasionally for certain situations (for example, in a game, I would poll the keyboard state every frame), but never in a loop that ONLY does polling, rather I would do polling as a check (has resource X changed? If yes, do something, otherwise process something else and check again later). Generally speaking though, I avoid polling in favor of asynchronous notifications.

The reasons being that I do not spend resources (CPU time, whatever) waiting for something to happen (especially if those resources could speed up that thing happening in the first place). The cases where I use polling, I don't sit idle waiting, I use the resources elsewhere, so it's a non-issue (for me, at least).

edited Nov 10 '10 at 20:14

Peter Mortensen
7,802 8 55 90

answered Nov 26 '08 at 11:26

Dan
4,582 3 29 40

If you are polling for changes to a file, then I agree that you should use the filesystem notifications that are available for when this happens, which are available in most operating systems now.

In a database you could trigger on update/insert and then call your external code to do something. However it might just be that you don't have a requirement for instant actions. For instance you might only need to get data from Database A to Database B on a different network within 15 minutes. Database B might not be accessible from Database A, so you end up doing the polling from, or as a standalone program running near, Database B.

Also, Polling is a very simple thing to program. It is often a first step implementation done when time constraints are short, and because it works well enough, it remains.

answered Nov 26 '08 at 11:01

JeeBee
12.5k 34 52

The thing about polling is that it works! Its reliable and simple to implement.

The costs of pooling can be high -- if you are scanning a database for changes every minute when there are only two changes a day you are consuming a lot of resources for a very small result.

However the problem with any notification technoligy is that they are much more complex to implement and not only can they be unreliable but (and this is a big BUT) you cannot easily

tell when they are not working.

So if you do drop polling for some other technoligy make sure it is usable by average programmers and is ultra reliable.

answered Nov 26 '08 at 11:03

James Anderson
19k 3 27 53

I agree with your point, in my current application am doing same polling mechanism and checking for file updates but it is very resource intensive and is hogging lot of resources, what would be an best optimal way to deal with the situation? – Rachel Feb 5 '12 at 20:22

I see many answers here, but I think the simplest answer is the answer it self:

Because is (usually) much more simple to code a polling loop than to make the infrastructure for callbacks.

Then, you get simpler code which if it turns out to be a bottleneck later can be easily understood and redesigned/refactored into something else.

answered Nov 26 '08 at 18:08

Ismael C
76 6

This is not answering your question. But realistically, especially in this "day and age" where processor cycles are cheap, and bandwidth is large, polling is actually a pretty good solution for some tasks.

The benefits are:

- Cheap
- Reliable
- Testable
- Flexible

answered Nov 26 '08 at 11:01

Ali Afshar
24k 3 62 89

I'm glad you qualified that with "some tasks" because the idea that we have endless processing power, space, or bandwidth is inherently floored. Eventually we will hit the limits of how fast processors can be, how much you can store in a 3.5in HDD, and how much data you can squeeze down an internet connection. This kind of thinking can't scale. – user109878 Jun 2 '09 at 1:43

I agree that avoiding polling is a good policy. However, In reference to Robert's post, I would say that the simplicity of polling can make it a better approach in instances where the issues mentioned here are not such a big problem, as the asynchronous approach is often considerably less readable and harder to maintain, not to mention the bugs that can creep in to its implementation.

answered Nov 26 '08 at 11:15

ljs
17.4k 22 85 116

As with everything, it depends. A large high-transaction system I work on currently uses a notification with SQL (A DLL loaded within SQL Server that is called by an extended SP from triggers on certain tables. The DLL then notifies other apps that there is work to do).

However we're moving away from this because we can practically guarantee that there will be work to do continuously. Therefore in order to reduce the complexity and actually speed things up a bit, the apps will process their work and immediately poll the DB again for new work. Should there be none it'll try again after a small interval.

This seems to work quicker and is much simpler. However, another part of the application which is much lower volume does not benefit from a speed increase using this method - unless the polling interval is very small, which leads to performance problems. So we're leaving it as is for this part. Therefore it's a good thing when it's appropriate, but everybody's needs are different.

answered Nov 26 '08 at 11:30

Valerion
726 5 14

Here is a good summary of relative merits of push and pull:
https://stpeter.im/index.php/2007/12/14/push-and-pull-in-application-architectures/

I wish I could summarize it further into this answer but some things are best left unabridged.

answered Aug 27 '09 at 8:45

bagheera
1,478 1 11 26
