DFD DEMYSTIFIED

DFD Demystified

As computer technology develops, so does the software which depends on this technology. Years ago computers were limited to a physical memory of 16 MB. This meant that instrument samples had to be rather small or limited. If you were lucky you could have five samples per instrument! As you can imagine, transposing one sample two octaves does not result in realistic sounding instruments. Thankfully those days are behind us and we no longer live with the limitations that kept samplers from realistic instrument playback using sample-based technology.

Nowadays it isn't unrealistic for a personal computer to accept up to 4 GB (or sometimes more) of physical memory. However, most of us are still stuck with that 1 or 2 GB max limitation whether it be a laptop for live gig or an older desktop. Whatever the case, there always seems to be the problem that there is never enough memory. As sampler instruments become more and more realistic, their physical size, measured in mega- or even gigabytes, also grows. Often, the physical size of all samples in one instrument exceeds the amount of available system memory. Even if you have 4GB installed in your machine, your operating system will allocate at least a few hundred megabytes (often more, and you slowly approach your limits of how many instruments you can load into physical memory. This is where DFD comes into play.

What is DFD?

DFD stands for "direct from disk" and is a technique for playing back large and very large instruments and samples without loading them entirely into RAM. In fact, only the first portion of each sample is loaded into RAM permanently; the rest is read from the computer's hard disk while playing the instrument. RAM is able to react virtually instantly, delivering the first portion of any sample the user requests, while the computer goes to fetch the next portion of that sample from the hard disk.

With DFD switched on, we can load samples with up to 2 Gigabytes each – and we can load quite a lot of them even with moderately equipped machines. Later, we'll learn how things work together, how much RAM we actually need, and how we have to set up our buffer sizes.

Theoretical background

It is beneficial to understand how streaming works if we want to achieve the best possible performance out of our hard- and software.

So why do we need RAM at all, if we can fetch things directly from the hard disk? The answer is: we can't. At least not as quickly as we'd need to. If we strike a note on our keyboard, we expect to hear sound immediately. To be precise, "immediately" is impossible in the universe as we know it because all sound takes time to travel through air, through converters, through cabling, and so on, but we still expect something which sounds "immediate" to our ears, which would be somewhere in the range of 1 – 10 milliseconds. You'll probably know this value – it's called "latency" and it also serves as a mark of how good your audio interface . Some of the best ones have latencies down to 1.5 msec). Ok, so we need to hear something a few milliseconds after the key is pressed. In short, there is no hard disk which is that quick (the very fastest disks available – designed to work in database servers – get down to about 4 msec). So the only way to solve the problem is to load the first portion of each sample in RAM (which *does* have the ability to deliver the sample virtually instantly) to give the hard disk a chance to fetch the next portion of the sample as it needs to be played.

Illustrative example for streaming one sample

Let's assume that you have a container which holds water. This container has a hole which allows the water to flow out of it on a continual basis. Using a bucket, your job is to fill the container with water so that it doesn't go empty.

If you are slow, then having more water in the container from the start will aid you in delivering the next bucket. With less water in the beginning, you will have to deliver the next bucket faster to keep the container from going empty.

However, the initial water level is not the only factor here. It would also follow that if the bucket were larger, you could carry more water and thus make less trips, with an obvious point that this bucket also takes longer to fill. On the contrary, a smaller bucket will require more trips, but fills quickly.

Add more buckets to the picture (one for each note played) and you'll quickly see how **speed** in general becomes a major factor.

How does this relate to DFD?

In the world of DFD, the container is your **DFD Voice memory**. Initial water levels can be associated with the **preload buffer**. The bucket is one **channel buffer**, while the bucket's capacity is your **channel buffer size**.

If these terms confuse you, then read on.

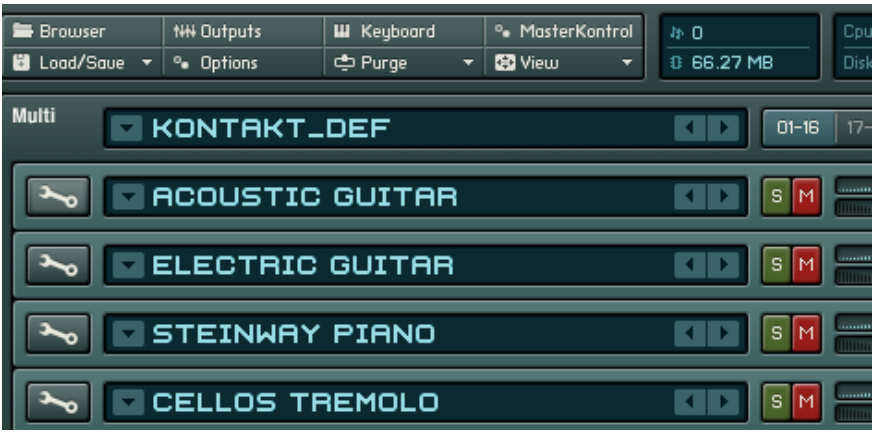
Technical background

The Preload Buffer

One preload buffer is necessary for every single sample in an instrument. This is because the performer may elect to play any key on the keyboard, and KONTAKT has no way of knowing which one it will be before it's actually played.

Consider an instrument made up of 200 distinct samples and a preload buffer size of 192kb; we'll end up with an instrument which requires 37.5 megabytes of preload buffer. Keep in mind, that the actual size of the samples does not matter. It's the same regardless of whether they are 1 Gigabyte or 200kb each.

in the above instrument you see the preload buffer uses 10.87 MB



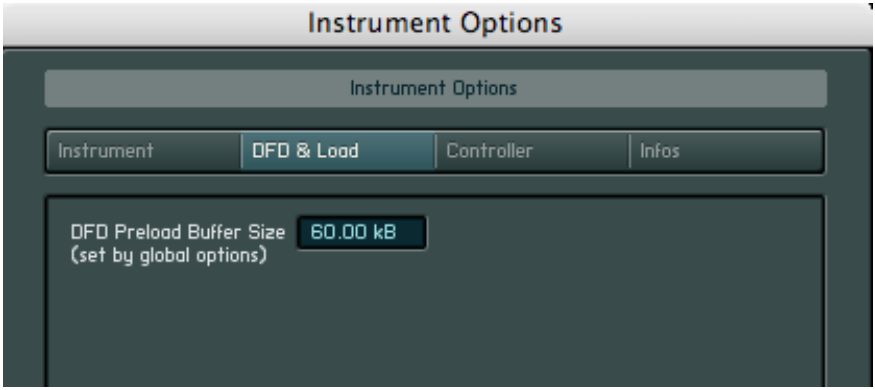
in KONTAKT's main display you see the preload memory total for all instruments in this instance.

So now we have our first simple formula for **used instrument memory**:

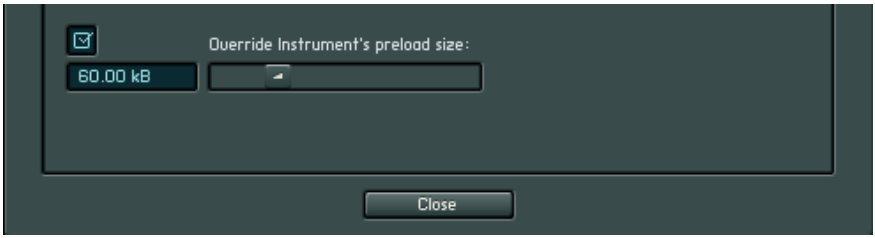
number of samples in an instrument * preload buffer size = used instrument memory

Translation: used instrument memory is the amount of RAM it takes to load the first portion of every sample in the instrument into RAM. Often, it's a small fraction of the total memory that would be taken up if every single *entire* sample was loaded into RAM.

When you load an instrument, preload RAM is allocated based on the instrument's preload buffer setting. This can be found (and adjusted as necessary) within the instrument's Instrument Options dialog.



Also, in KONTAKT 2.1, you may override individual instruments' settings with a global preload buffer setting.

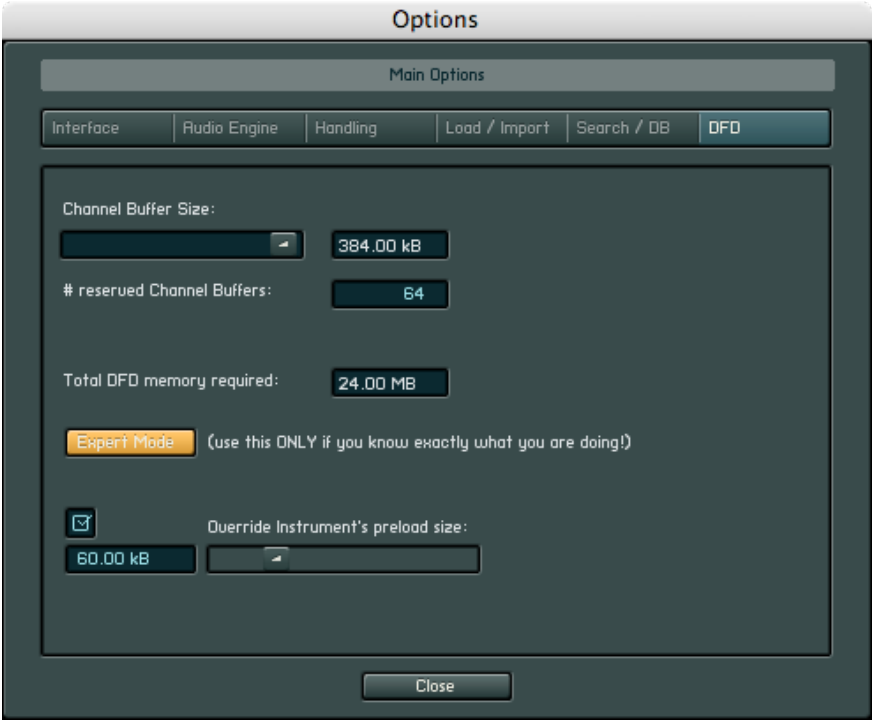


The preload buffer setting is specifying *how much* of each sample you want to preload into RAM. As the preloaded portion of the sample(s) being played is playing back, the hard disk has already begun fetching the next portion of the sample(s) being played. This amount of time (often a fraction of a second) depends directly on the amount of memory we reserve for the first part of the sample (the preload buffer size).If that buffer is large, it is sufficient to play back for a longer time, which makes the hard disk's job much easier.

If that buffer is short, the hard disk will have to work harder and quicker, but will save more RAM. So obviously there is a tradeoff here. On one hand it's possible to use relaxed settings which are easy on the hard drive but take up larger amounts of RAM. On the other hand, it's possible to use such short preload times that the hard drive isn't able to deliver the samples fast enough. Polyphony also figures in; if the hard drive has to fetch a few samples at a time, it may work fine, but if it has to fetch hundreds of samples at the same time (for example, in a full orchestral sample library), it may get tricky. Every users' hardware and performance needs are different, so there are no one-size-fits-all answers. However, we have some more guidelines to share.

The Channel Buffer

In order to continue a voice, the DFD system will continually fetch the next buffer for a voice and put it into a **dedicated RAM area**, from which KONTAKT will read the samples. The dedicated RAM area (the container) is reserved by KONTAKT at startup and its size is based on two DFD settings:



channel buffer size **and** # reserved channel buffers

Since these buffers are voice dependent, and one voice in KONTAKT can occupy up to 16 channels, they are called **channel buffers**.

How many channel buffers do we need?

That's relatively easy: it depends on how many voices we plan to play simultaneously. If we have a maximum polyphony of – say – 100 voices, we need 100 of these buffers. The size of each channel buffer (called **channel buffer size**) will determine how quickly the next portion of the sample is fetched.

Note: A stereo voice will use 2 of these buffers, while a 5.1 voice will use 6 of them. This is an important factor when setting up the number of reserved channel buffers.

Knowing this we arrive at another formula which accounts for **DFD voice memory** (Engine tab):

reserved channel buffers x channel buffer size = **Total DFD memory**

It is important not to confuse this memory total with the memory usage display in the KONTAKT instrument panel for example. They are two values which in part make up for the amount of system memory KONTAKT will allocate when running with loaded instruments.

Putting it all together

Now that we know almost everything about the theoretical basics behind DFD, we come to the following conclusions as far as memory is concerned:

1. We need a bunch of preload buffers, as many as we have samples loaded.
2. We need channel buffers, precisely as many as the voices we want to play.
3. We can adjust each of the buffers' sizes independently to match our requirements and memory.

Additionally, it's important to remember that:

- Channel buffers are always there, regardless if anything's loaded or not.
- Preload buffers can become quite numerous, if the loaded instrument(s) contain many samples. Preload memory consumption derives directly from the number of samples used, so it might be necessary to set the preload buffer size quite low, if the loaded instruments are large in terms of sample count and installed memory is limited.

- DFD voice memory is allocated on a per-instance basis. It is shared by all the instruments loaded into one instance of KONTAKT.

Combining the illustrative example and our theoretical knowledge would allow us to derive at the following:

The larger the bucket and the more voices being played, the longer it will take to fill the buckets with water, i.e. fetch the next channel buffers. If you want to play lots of notes (high polyphony), this could mean that filling the buckets for all these notes might take so long that it will not be done in time. This could result in a DFD overload.

Naturally, it might help to reduce the bucket (channel buffer) size here, but there is a limit: the smaller the bucket, the earlier the next channel buffers have to be fetched again. In this case, the **seek time** of the hard drive will have more effect on the performance of the DFD.

Other considerations

Obviously, with the vast amount of system configurations and individual needs we cannot give a "magic number;" however here are some general guidelines which should help to get you achieving better performance on your computer.

Hardware Basics

The hard drive

Of course, The most critical part is the hard disk, and obviously we can gain a lot by choosing the fastest hard disk we can get. "Fast" in the hard disk world has basically two dimensions: transfer bandwidth and access time.

Although transfer bandwidth is an important factor, access time can be even more important. Current 3.5 inch desktop hard disks get values of about 40 megabytes transfer bandwidth per second and an average access time of about 10 ms. 40 MBs transfer bandwidth would be sufficient for about 200 – 250 stereo voices @ 44.1kHz / 16Bit. But unfortunately, there's still the access time to consider. Access time is the average time that is needed to address a randomly chosen target position on the hard disk (from any other previous position). And in fact, that's exactly what is happening when it comes to DFD: whenever several voices are playing simultaneously, many different (virtually randomly spread) data portions must be retrieved as quickly as possible.

Another point about access time is the fact that it cannot be scaled, as can be the transfer rate. By using cheap and popular RAID arrays, the transfer rate can easily be raised to almost any degree, but access time will stay approximately the same, even if we combine 4 or more drives. So we come to the following, very important conclusion: If you plan to use large libraries, look for a hard disk with a really good access time.

Most desktop disk drives, as stated previously, are 7200 rpm devices and deliver access times of about 10 msec. But if you want to get high polyphony (150 – 250 stereo voices) from large libraries, that may not be good enough. In such cases, you'll need a faster drive. Such drives are used for example in database servers and usually run at 10,000 or even 15,000 rpm (and they are more expensive, unfortunately). These drives achieve access times down to 4 – 5 msec. The Western Digital "Raptor" is an example of a modern 10,000 rpm drive with such access times. If you are a high-end user and use huge libraries and high polyphony, you'll probably want to use a drive like this – or even two of them in a RAID configuration.

Hard disk interface

Nowadays there's hardly any difference (for streaming purposes) between interface standards like ATA, SCSI, FireWire or SATA. They are all able to transmit the data faster than any hard disk can deliver the data. Still, there might be differences in the way the hard disk is connected: the best possible performance is generally achieved with the disk being attached to the main internal connector. That does not mean that an external Firewire disk will not function as well, but there are cases where that would be true: on the PC side, Firewire (or RAID) controllers are often attached to the PCI bus, which also holds the sound card. Cheap adapters and/or bad drivers can in some situations have a bad influence on DFD performance and produce sonic artifacts. Use high-quality adapters and put drives on dedicated busses whenever possible.

Buffer sizes

The next topic is buffer sizes: we've seen in the above theoretical example that larger buffers can relieve pressure from the hard disk, hence improving both performance and polyphony. In fact, large buffers can (to some extent) compensate for slow access times. It's not a simple "more buffer, more voices" relationship. There are situations where raising buffer sizes helps performance to a certain point and then decreases afterwards. A "best for everyone" setting does not exist, so a little research is still required for each individual situation.

Memory

We've seen that we need quite a lot of memory, and our memory requirements increase substantially if we want to use instruments with many samples. We'll end up with many preload buffers, and they can add up considerably. The OS uses RAM, as does your host sequencer and any other programs, plug-ins, and various extensions you may be running.

How much memory do we need? The answer would be, "the more, the better," especially when it comes to large libraries. RAM is a crucial resource. Again, it largely depends on what instruments you want to use. The DFD control panel will show you how much total DFD memory your channel buffer configuration takes and in the instrument header you'll see how much the single instruments take. Adding this will give you an impression. Of course there is a lot of additional demand for RAM (the operating system itself, the sequencer, other software instruments, etc.). As a hint for memory requirements we could state the following:

- 512 Megabytes can be considered minimum for any decent audio system
- 1 Gigabyte is a good value for a powerful all round system and should serve well for most standard DFD applications
- 2 Gigabyte would make a high-end system useable with large and high-end libraries.
- 4 Gigabyte is actually the limit for any 32 bit application. To this date nooperating system even allows a single application this total amount.

Last, but not least...

Know what resources your computer has available

Ok, so you aren't a technical wizard. It doesn't matter. If you are using a computer for music, then you should know at least the basics. That includes knowing how to diagnose available system resources.

Both PC and Mac utilize an advanced virtual memory handling system which can cause sample playback issues if available memory is not monitored correctly.

PC: control-alt-delete, opens the Windows Task Manager. Here you can see the system-wide memory usage, including individual program amounts.

Mac: Applications / Utilities / Activity Manager. Here you can see how much free memory is available to KONTAKT. For more on this topic please check our online OS X tutorial. **http://www.nativeinstruments.de/index.php?id=niosxtut_us**

Experiment with various settings

For those of you have no playback problems, then this shouldn't apply to you. If you experience any sort of playback trouble, then the DFD settings could be responsible. Keep in mind that things in this area are very hardware contingent. The only way you are going to find the "sweet spot" for your computer is by trial and error, and remember: if it isn't broken, don't try and fix it.

1. Check the instrument preload buffer.

If your hard disk is slower, around 4200 or 5400 rpm, then increasing the preload buffer could make a difference. This will give DFD more headroom to do its job. It will also require more RAM to get the job done. Remember this is on a -per sample- basis. So if your thousand layer Grand Piano has a 240kb preload buffer, then you will lose RAM pretty quickly, just by loading the instrument.

2. Determine how many voices you need to play.

You should know roughly how many voices you wish to get out of KONTAKT. This isn't necessarily a must for the beginning of your projects, but as things develop you may want to adjust DFD settings according to your project size and voice count. Now that you know what the DFD settings mean, take a look at them. Do you really need to playback 1024 stereo voices in one instance? Just because this option exists doesn't mean that your setup can handle it. The KONTAKT engine is prepared to handle such voice counts, but only if the computer can provide the resources. Keep in mind that if you require more voices than set reserved channel buffers, you will get errors.

3. More is not better.

A collective mistake is often made by users who think that more is better. While this may be true for lots of things in life, it is not for DFD settings.

Whether you are in the Easy or Expert mode, be aware that moving the sliders to the extreme right WILL allocate a lot of system memory to DFD. If you have that extra memory, great! If things are tight, then this is especially troublesome in the event that this amount of memory is not available to KONTAKT. The best thing to do is move the slider a bit to the right, then try playback at your desired polyphony. For large projects, with more KONTAKT instances, DFD Voice memory is allocated on a **per instance** basis. This means that if you assign 480 MB to DFD in Easy mode, 2 KONTAKT instances will eat almost 1 GB. This, even before loading one instrument!

Conclusion

Well, that's it. You made it through the KONTAKT DFD tutorial. It isn't a light topic, but we hope you are better informed about this mysterious KONTAKT feature that is designed to allow you to load many times more instruments at once than you would be able to if everything was simply loaded into RAM.