Build a render farm

on the web
Additional material for this article is in the Stop Press section of our site
www.3dworldmag.com

Don't send your old PCs to the dump: use them to build a distributed network to speed up your renders. We run through the tools and know-how you'll need

BY GRAHAM MORRISON

Ten years ago, distributed computing would have been an impossible dream for the average home users. But now, older machines and cheap beige boxes can be used to create a powerful rendering farm without either a massive financial outlay or a attaining a degree in computer science. It can still be complicated, but the rewards are worth the effort.

There are two widely used models for distributed computing. The first has the best name: a Beowulf cluster. (If that doesn't scare you off, nothing will.) The idea is simple, even if the implementation is a little tricky. It's basically a network of computers that act like a single machine. The operating system needs to be tricked into thinking that all distributed components, including memory, storage and processing, are all parts of a single device. The operating system then manages tasks as it would on a multi-core CPU, juggling execution from one processor to another while remaining oblivious to the fact that signals that would normally be sent over minuscule bits of hardware inside a single computer are now making their way over a local network.

The advantage to this approach is that any piece of software designed for a multi-CPU environment will work, as the software thinks it's running on an ordinary operating system. The disadvantage is that this usually means your only choice for that operating system is Linux, and there aren't many rendering applications that will work with a Linux Beowulf cluster.

By far the more common variety of render farm is a normal computer cluster: a group of interconnected machines, each with its own operating system and resources. The machines just need to be told what to do, which is the job of the queue manager. Before it can start queuing jobs, the principal rendering task needs to be split into smaller separate jobs – one for each computer to work on.

The simplest option is to give each computer on the network a single frame of animation, so that they can be treated completely independently. Each rendering machine can then work alone on a frame. Alternatively, some queue managers support bucket-based distribution, splitting a single frame up to send different sections of the same image to different machines. The queue manager will reconstruct the frame after all the jobs have completed. Either way, the result is a dramatic improvement in total rendering times.

> Using old beige boxes to set up a render farm takes patience, but the rewards are well worth the effort

The term 'farm' is a surprisingly accurate description of what actually happens in a render network: processing power is farmed out to machines connected together on a local network. The machines need to be local to each other, because only a physical connection can provide the bandwidth to transfer the vast amounts of data required to render a scene. Unless you've got the money to spend on fibre channel, this means good old Ethernet. Nearly every machine from the last five years will include the RJ-45 port used to

NETWORK STRUCTURE

A rendering network consists of a number of basic elements. Here's a quick run-down of what you need and where to put it

RENDER NODES

Each node of your render farm consists of a bare-bones PC. All you need is a processor, a small hard drive, some memory, and a network connection for the hardware. The software you use is dependent on your rendering application. Be prepared to pay a licence fee for additional nodes on your network if you're using *XSI*, or rendering with *mental ray* in *Maya*, for example.

NETWORK HUB

A network hub, or switch, bridges the connection between your rendering nodes and the server that controls the render queue. Network configuration is handled by the server, but you should reconfigure it so that you assign IP addresses manually, rather than the server doing it automatically. This way, each node takes the same address every time it's booted rather than a different one.

ATTACHED STORAGE

Network Attached Storage is the easiest storage solution for a render farm because it can plug directly into the network switch. This makes it accessible by all machines on the network, and secure behind any firewall running on your server. An alternative is to use storage on the server machine.

THE SERVER

The server can be either a low-powered machine used to control the render queue, or your own desktop. If it's your desktop, you can offload rendering duties to the farm while still using your machine for editing your project. Running the render queue doesn't take up too many system resources.

connect your computer to an Ethernet network. Many newer machines will support the fastest 1000BaseT Gigabit standard, which is capable of transferring a 500MB image in less than half a second. That might sound fast, but when you consider that you need to send images back and forth between every machine connected to the network, speed isn't a luxury, but a necessity.

Even if you're stuck with an older network card, there's no need to upgrade straight away. The previous generation of network technology is around ten times slower, as denoted in the name 100BaseT Ethernet, but this is still adequate for smaller projects. If you find that transferring data becomes a bottleneck to better performance, you can always upgrade your networking hardware. Gigabit expansion cards can be bought for less than £20.

Network configuration is considered something of a dark art, but there really isn't all that much to it. Physically joining your machines together is the easy part: just connect each to a network switch or hub, as shown in the diagram above. The switch needs to be capable of the same transfer rates as your computers, otherwise the entire network will default to the lower speed.

One machine will typically act as the master for the other units connected. This is known as the server, and each machine connecting to it becomes a client. Most queue managers require a server to run the application that controls each client. Each client is assigned a job from the server and reports back to it when the job is complete. Project data is stored on a shared drive, so that each machine has access to the same resources.

If you're using second-hand hardware, be aware that while anything faster than 1.4GHz can make a worthwhile contribution to your rendering set-up, older machines quickly become obsolete. Even if your equipment cost thousands five years ago, it might be more cost-effective simply to buy or upgrade to current hardware. A top-of-the-range 1.4GHz CPU from 2002 can now be bettered by a mid-range 3GHz Core 2 Duo, and that doesn't take into account advances in memory and system bus speed. Generic PCs are incredibly affordable, and you can now buy a fast machine with 2GB of RAM for a few hundred dollars.

If you go to the trouble of buying a few cheap PCs for the sole purpose of creating a render farm, you need to give priority to processor speed and the amount of RAM. Both have a dramatic effect on performance, and it's no good having plenty of one and little of the other.

However, it's not worth spending money on a large hard drive. You will need a massive amount of storage – but then, it's likely that you already use some form of network attached storage (NAS) device, and this needs to be accessible by all machines on the network. Alternatively, use a server with a RAID setup and plenty of capacity. You can even save a few pounds by excluding the optical drive from each networked machine, although this can be more trouble than it's worth.

Finally, you won't need a monitor, a keyboard or a mouse for each new machine. These are only needed while you install the operating system, so you'll only need a single set to use while configuring your farm. After the operating system is installed, you can use a tool such as *Remote Desktop* or a free VNC application to remotely control each machine in your network. Your networked computers will then appear in a separate window in your desktop, taking control of your mouse and keyboard whenever the window is selected.

TECH FOCUS

QUEUE MANAGERS

For a larger network, you may need a third-party queue manager. There are a number on the market, but here are five popular ones

Some rendering application developers have been a little slow to build distributed computing support into their own applications, which has led to a glut of third-party queue management applications. Many add functionality that just doesn't exist in the main application (such as *Blender*), but a queue manager is also the best way to get a good overview of your render farm's performance.

BUTTERFLYNETRENDERER
www.liquiddrreamsolutions.com
From $45
Windows, Mac OS X, Linux
LightWave 3D, Maya, 3ds Max, messiah:studio
A popular choice that's grown from just *LightWave 3D* support to support *Maya* and *3ds Max*

DRQUEUE
www.drqueue.org
Free
Windows, Mac OS X, Linux
***Blender*, configurable to other 3D applications**
Open-source advocates love *DrQueue*. It's powerful but it's complicated to set up

FARMERJOE
http://blender.formworks.co.nz
Free
Windows, Mac OS X, Linux
Blender
One of the easiest ways of getting into render farming. Easy to install and use

MULE
www.epicsoft.net
From $75
Windows
LightWave 3D
***LightWave* users get into a frenzy for this hard-working network rendering tool**

QUBE!
www.pipelinefx.com
Contact reseller for pricing
Windows, Mac OS X, Linux
3ds Max, After Effects, Fusion, Maya, mental ray, Shake, XSI
Massively powerful, and with a price to match, but it can be made to work at almost any scale

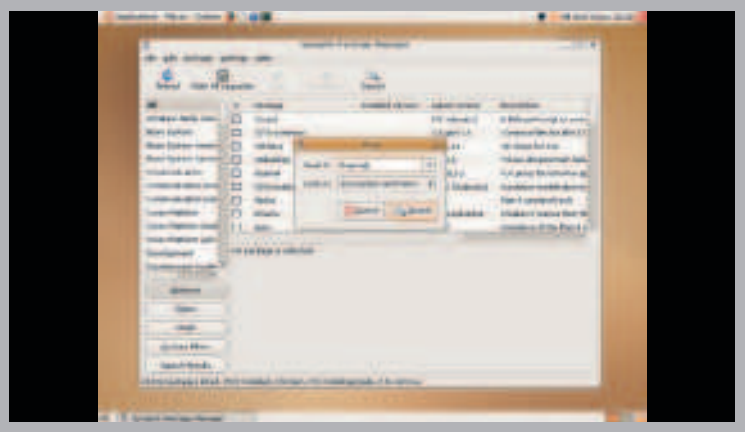
Many online retailers will let you buy a bare-bones system to your own specification, and if you're able to buy more than one machine with the same spec, installing the operating system and your applications becomes much easier. The process is as simple as installing everything you need onto a single machine, making an image of that machine using a tool like *Norton Ghost* and restoring this copy to each new machine on your network.

But which operating system do you choose? The choice divides between commercial alternatives and the open source operating system Linux. Windows or Mac OS X will feel familiar, but their cost can be a significant proportion of cheap hardware. This becomes important when your network starts to grow from just a couple of machines to five, 10 or even 20 computers. In contrast, Linux is available for free, and you can install the same copy as many times as you want. It's also secure and stable, but it can be unintuitive for users without any prior experience.

Your choice of OS is also going to be dictated by your rendering software. *3ds Max* is not available on Mac OS X, although other major programs are. There is no Linux version of *3ds Max*, *Cinema 4D* or *LightWave 3D*, for example – although the network rendering module for *LightWave* does run on Linux – but there are native versions of *Maya* and *XSI*.

A render farm also needs more than application availability: your software needs explicit support for distributed rendering. This varies from application to application. For example, a single licence of *3ds Max* or *LightWave 3D* enables you to network-render across a functionally unlimited number of machines. *Maya* does the same, but only for certain render engines: *mental ray* rendering is restricted to eight additional CPUs in *Maya Unlimited*, and two in *Maya Complete*. If you're an *XSI* user, you really need *XSI Advanced* to take advantage of network rendering: the entry-level *Foundation* edition of the software only enables you to render from the user interface.

A render farm can grow in stages. You could even start with a single machine connected to your desktop

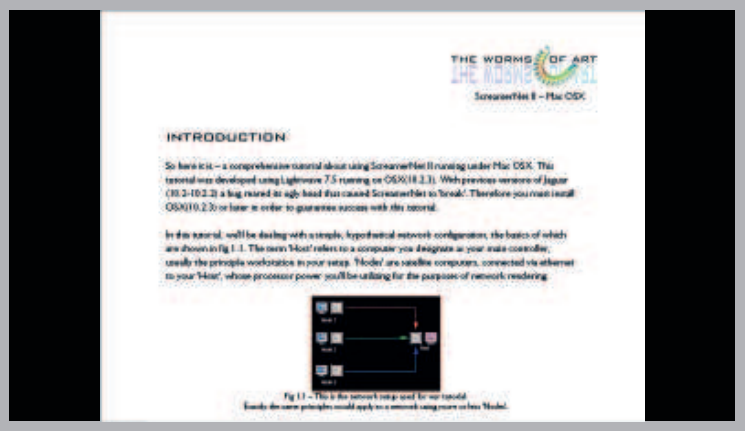


THIRD-PARTY CONTROL If you're serious about your rendering network, you might want to consider a third-party queue manager. *DrQueue* (pictured above) is a powerful application, but it's difficult to install on a Windows machine. If you use Linux, you often find the software pre-installed, however

You might also want to consider a third-party queue manager (see above). A queue manager isn't a replacement for the renderer itself: it just manages the resources for your project and launches a rendering process for each application running on the farm. One advantage to this is that you can use a manager in a mixed-application environment, allowing animators to submit jobs from different renderers. A queue manager will also often let you give different priorities to different tasks as well as juggle projects between machines. But the biggest advantage is that animators have complete control over the render farm. They can add, remove, pause and play jobs, as well as change the priority of those already running from a single application.

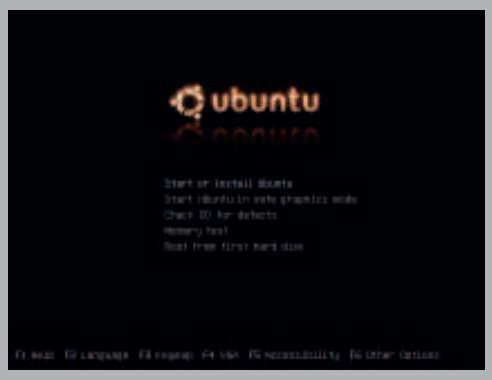
All this information might seem intimidating, but the good thing about building your own render farm is that you can do it in stages. You could even start with a single machine, running a crossover network cable from your main desktop to add a little horsepower. If you find yourself farming out more processing to the extra CPU while you carry on working, adding another machine or two is easy. This is perhaps the biggest advantage to using a render farm: it can grow to accommodate new projects. And when things are quiet, they're great for the occasional game of *Unreal Tournament*...

THINKING OPEN SOURCE *Blender* is a good choice of 3D application for render farm experiments because it's free to use, and widely used with distributed computing. It also helps that there are versions for Windows, OS X and Linux, expanding your options when building a rendering network

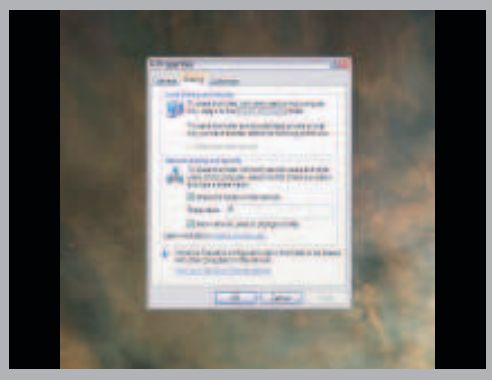


TUTORIAL SUPPORT Network rendering options vary from application to application. *LightWave 3D*'s ScreamerNet module enables you to render across a near-infinite number of CPUs. You can find a good tutorial at **www.the-worms-of-art.com**. It's for *LightWave 7.5*, but should work with later versions

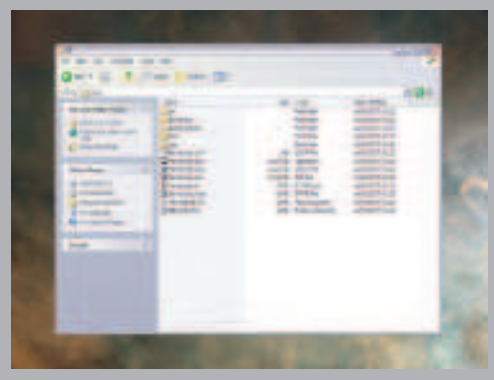
STEP BY STEP / Set up a free render network using Farmerjoe and Blender



01 We're going to use a Windows machine as the server, and a couple of spare PCs running Linux as the rendering clients. These days, Linux is easy to install. The most popular distribution is Ubuntu, which features a Live CD installation procedure that's smoother to run than Windows. Just put the disc on your machine and reboot. Installation is just a couple of clicks on the wizard.



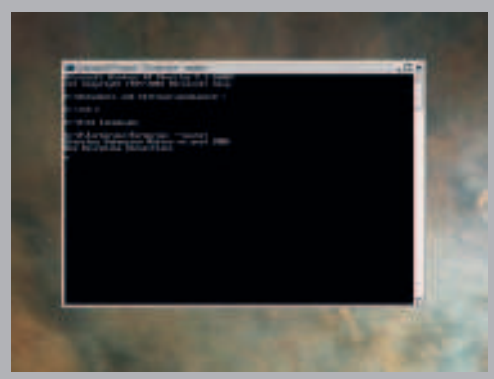
02 Back on the Windows machine, we need to share the storage device. For a local hard drive, that means enabling file and printer sharing in the Windows network control panel. You will also need to enable sharing for the specific directory you want to use for data storage. Both Linux clients will then be able to read and write from this drive. Full-sized versions of the screenshots accompanying this walkthrough are included in the supporting material for the article, which can be downloaded from the Stop Press section of the *3D World* website: **www.3dworldmag.com/stoppress**



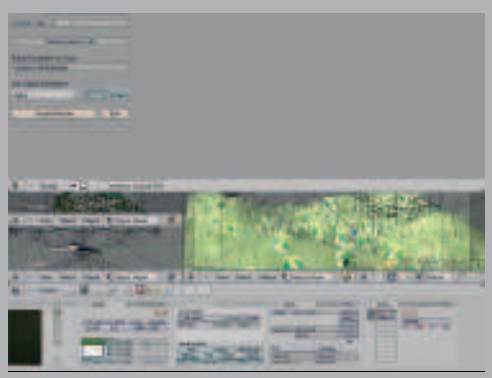
03 The queue manager we're using is *Farmerjoe* (free from **http://blender.formworks.co.nz**) which expects a specific file structure in the root of the shared drive. This consists of the program executable (for Max OS X, Windows or Linux) as well as 'jobs' and 'logs' directories for when the server is running. It also expects to find a copy of *Blender* in the 'bin' directory.

04 The shared drive needs to be mounted onto your Linux machines so that they can read and write to the same drive. The command for doing this is a little technical, but well documented in Ubuntu. Network storage in Linux appears just like local storage, and will become part of the normal directory tree.

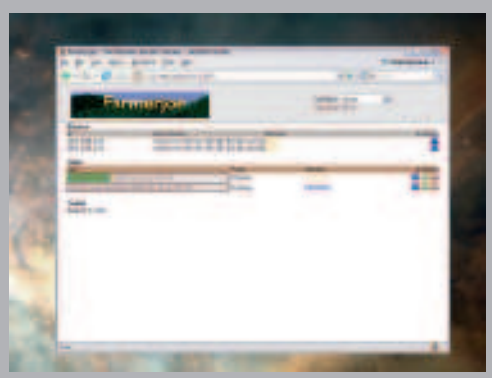
05 Open 'farmerjoe.conf' in *WordPad* or another text editor: this is the configuration file for both the clients and the server. You will need to change the IP address of the server, as well as the locations for the shared storage. Most other queue managers take the same approach, but this file is tame in comparison to some of the others. Potential *DrQueue* users should consider themselves warned! Fortunately, good user tutorials for many queue managers do exist, and can be located with a little ingenuity and diligent use of Google. Failing that, you may need to call on more experienced users in the developer's forums.



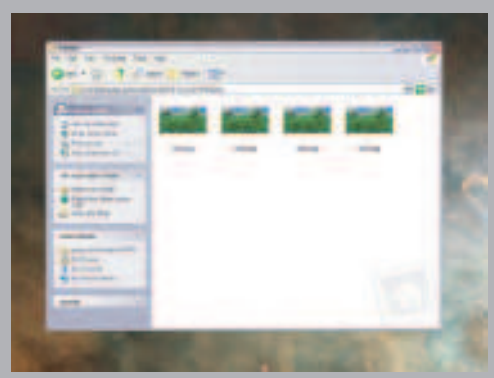
06 The next step is to run the server and any Linux clients you want to use. In Windows, you need to open Start > All Programs > Accessories > Command Prompt, change to the shared drive, and type 'Farmerjoe --master'. In Linux, you also need to open a command prompt (called a terminal) and just run 'farmerjoe.linux'. The clients will find the server and connect automatically.



07 Now launch *Blender* on Windows and load the project you want to render. You send the project to the render farm with a script: you can either load this as a text file or add it to the Blender Scripts directory. After it's running, just click on the Submit Render button. This will send the entire project to the *Farmerjoe* rendering application.



08 You can check on the progress of the jobs you've sent by opening a web browser and pointing it to **http://localhost:2007**. You will need to have run 'Farmerjoe --appserver' on the Windows machine first. It loads a small web server that will update with the progress of each job, as well as let you pause, change their priority and manage the clients that are connected to the server.



09 Each completed frame is helpfully saved into the 'jobs' directory on the shared drive, and each job will have its own directory for images. This is where the clients that are connected to the render farm are placing the fruits of their labour, and where you can harvest those frames. And that's it: you should now have one fully configured render network, and all without having had to spend a cent.