

Snake Dance

Megastreaming with Python

Daniel James talks to the developers of the Kamaelia streaming media system

In the first century of radio, even the national broadcasters only had to provide a handful of channels each, delivering media in a serial fashion. Now that the internet has enabled programmes to be streamed on demand, the massively parallel load on networks is growing at a rate which could be unsustainable if broadcasters rely on existing server technology. In the UK, the BBC is already serving millions of streams per day, and has over half a million hours of audio in its archives which could potentially be opened up for internet delivery.

The BBC's research and development department is now working on Kamaelia, a streaming platform designed to scale up for the future. The software is written largely in Python, running on Linux, OS X, Windows and the Symbian-based Series 60 mobile phones. Kamaelia is released under three free software licences - the Mozilla Public Licence and the GNU LGPL as well as the GPL. This combination has been chosen in order that the code can be linked into as many other projects as possible.

Michael Sparks and Matt Hammond are two of the key contributors to Kamaelia at BBC R&D, and in this interview they speak on behalf of the Kamaelia project, rather than for the R&D department or the BBC as a whole. Before arriving at the BBC, Sparks previously worked on web caching for JANET, the UK's higher education network, and then moved to Inktomi, now part of Yahoo! "I've been using free software for ten years now at least. During my stint at university I needed a C compiler, and had been using DJGPP largely because whilst I

didn't have much cash, and people did have pirated copies of Microsoft and Borland C, I didn't particularly want to rip off their software. Using DJGPP led to me trying mini-linux - a four-floppy Linux distro that worked on my 4MB 386 at the time. I found it worked nicely and moved onto Slackware."

"The case for Kamaelia was based around the idea that in order to help develop open standards for network delivery in the future, a platform was needed for experimentation. The long history surrounding network protocols being developed in what we now call open source or free software was also part of the deciding factor. After all, the TCP/IP stack and earliest email implementations predate the term free software, whereas the web, usenet and IRC all predate the term open source. Back then it was simply scientific and engineering best practice."

"We needed to have a system that made it easy to collaborate with others, a low barrier to entry, allow people to reproduce (or refute) our results, to experiment with new protocols and do so under potentially very high loads. At the time we didn't see any systems, open or proprietary, that matched those criteria - and in order to allow the collaboration aspect, we released as open source."

Hammond joined BBC R&D after graduating from the University of York. He also believes that allowing the Kamaelia source code to be released is a pragmatic move. "If this were a more traditional broadcasting technology project, such as the development of digital

television standards, then we would be looking to collaborate through an industry forum such as the European Broadcasting Union. However with Kamaelia we're talking about technologies for the internet - which is a very different environment. Development and standardisation happen through collaboration and consensus, rather than through closed industry fora. Developing as an open source project is more attuned to this way of working. It is also a way to build interest, and hopefully momentum, around something that we think is a worthwhile technology. With any luck people will benefit, not just from any potential improvements in how the BBC serves content, but also from others utilising the same technologies."

CONCURRENCY

With multi-cored processors likely to become common in listener homes before long, Kamaelia has been designed from the ground up for concurrent processing. The concurrency subsystem in Kamaelia is called Axon, and is again designed for ease of use, as Sparks explains: "Our key analogy is to think of computer systems as workers in offices. Consider someone working at a desk with some old fashioned wooden inboxes and outboxes. You can have a component that reads from files, another that serves to a network connection. Tell a 'postman' to take messages from the file reader and deliver them to the network server, and you have a simple server. Similarly if you have components that can act as network clients, decoders, and audio players, then

hooking these together can give you a simplistic network audio client."

"This allows you to focus on small, targeted components and build interesting systems. This should sound familiar since it's the same idea behind Unix pipelines. The difference is that our data can be any Python data object rather than flat file-like data. Since components only deliver to outboxes, receive from inboxes and interesting systems are formed by asking a postman to do deliveries, there is nothing stopping the system from placing components on multiple CPUs or on different systems. Until those systems come on line, the fact that we also target working on a single CPU means that we can deal with naturally concurrent systems, such as serving content to many clients simultaneously."

Hammond points out that for the time being, Axon runs mainly in a single thread. "It uses co-routineing, in the form of Python generators, to switch between tasks. A generator is a piece of code with 'yield' statements interspersed through it. These are the points at which control is returned to the caller. When the generator is called again, it picks up from where it left off. This is a nice piece of syntax that lets you write sequential-looking code, that in actual fact is implemented as a state machine. A scheduler

likes, dislikes, and whether you like that information to be shared or not, and helps interact with other Kamaelia systems in close proximity. These may be running on almost any hardware, due to Python's portability."

"Peer to peer systems are interesting from a pure technology perspective. They tend to be able to handle issues like distributed mesh setup, efficient delivery of content from multiple sources, and can be viewed as a highly effective distributed caching system. Common issues with P2P tend to revolve around trust and verification of content, and start-up speed. Multicast is by contrast an old technology, with the earliest RFCs dating back to the late 80s. It is often misrepresented for ease of discussion as send once, receive many. In practice however, it's more like many senders, many receivers. Multicast is becoming interesting because many companies run it inside their networks, and we have arranged private peering with ISPs for a simplified multicast system. Nevertheless, the real multicast world today consists of many small islands of connectivity."

"If we look at these two systems we can clearly see that the weaknesses of one are the strengths of the other. P2P can be slow to start up, but if that start-up occurs over multicast, rather than having one source

allowing for concepts from P2P like supernodes to act as large scale caches, which can be implemented at the discretion of ISPs."

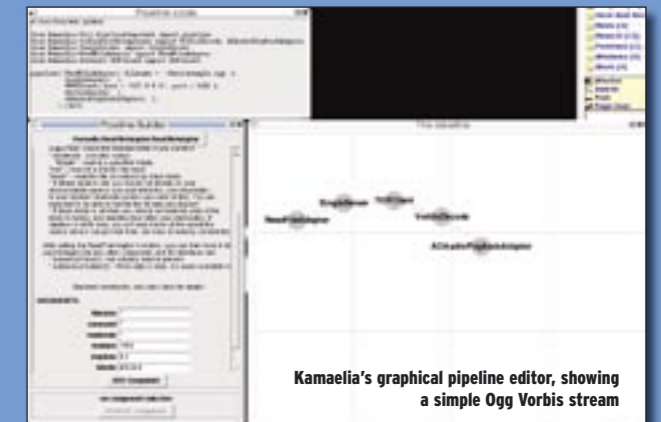
GETTING INVOLVED

The Kamaelia code is available on SourceForge, together with documentation and other material. According to Sparks, the technology is ready to be incorporated into third-party development work. "If you have a problem you want to solve which involves networks, audio or video, consider looking at Kamaelia to see if there's code you can reuse. We're in the process of compiling a list of potential projects people might want to implement, as well as projects we will be working on and need help with."

"It is likely that in the long term we will have highly optimised components written in C++, and having a native version would be extremely useful to avoid large algorithm changes. Similarly we'd be very interested in seeing more work with Kamaelia in embedded systems. The Series 60 port is a start, but just that. For Kamaelia to really fulfil its potential, being available on many platforms will enable users to join their disparate devices together using a simple coherent set of tools."

As Hammond puts it, the best way to get involved with the project is simply to download

Kamaelia has been designed from the ground up for concurrent processing



can then go round the tasks that need executing, calling each one in turn. This has massive efficiency benefits on a single processor as the overhead of switching between tasks is much lower than with processes or threads. You could quite happily have tens of thousands of concurrent tasks implemented using co-routineing. With threading you would be lucky to achieve an order of magnitude less."

P2P MEETS MULTICAST

Although Kamaelia does not yet have many components designed for peer to peer networks, Sparks believes the technology has great potential when combined with multicast delivery. "We can envisage Kamaelia code running on your mobile phone that learns your

coming on line after another, you could have thousands all at once. Similarly, P2P's ability to simply and easily create meshes can be used to join multicast islands together. This can be done using Kamaelia today - the next step is to integrate the mesh creation code with the tunnelling code."

"When you look at the system this way, multicast can become a large scale network broadcast system, usable by many people - not just the BBC - using P2P as a wide area multicast routing system. Furthermore, the P2P mesh can also be used as a large scale network cache - a massive distributed PVR if you will. This goes further than the traditional content distribution systems by pushing the entire cache right out to the very edges of the network, whilst still

and play with the code. "Try turning routines and libraries into components. There are tutorials on the Kamaelia website and blog that show how easy this can be. We'd love to get feedback from anyone trying to use Axon and Kamaelia themselves, for whatever purpose."



key links

The Kamaelia project
kamaelia.sourceforge.net

BBC Research & Development
www.bbc.co.uk/rd/