Real-time sensor signal capture from a harsh environment

Mark Purcell
IBM Research - Ireland
mark_purcell@ie.ibm.com

Aravind Vasudevan
Trinity College Dublin
vasudeva@scss.tcd.ie

David Gregg
Trinity College Dublin
dgregg@scss.tcd.ie

***Abstract*—Understanding the baseline underwater acoustic signature of an offshore location is a necessary, early step in formulating an environmental impact assessment of wave energy conversion devices. But in order to even begin this understanding, infrastructure must be deployed to capture raw acoustic signals for an extended period of time. This infrastructure is comprised of at least four distinct components. Firstly, a hydrophone, deployed underwater, which is capable of operating at a high sampling rate: 500,000 16–bit samples per second. Secondly, an analog/digital converter (ADC), to which the hydrophone transmits raw voltages. Thirdly, a communications infrastructure for bridging the gap from the ADC to shore. And finally, an onshore base-station for receiving the signals and presenting them to a remote analytic or simulation infrastructure for further processing.**

Attempting this signal capture in real-time poses many problems. On a practical level, deploying cabled infrastructure to deliver power and communications to the offshore components may be prohibitively expensive. However, reliance on solar power may result in interruptions to real-time wireless transmission. Additionally, a high sampling rate will require significant base-station memory/storage/processing capabilities as well as potentially high costs of delivery to a remote infrastructure, part of which could be alleviated by real-time signal compression. This paper discusses our attempts at implementing such a system which would reliably acquire real-time data and scale with growing demands.

***Keywords*-Real-Time; Data Acquisition Infrastructure;**

I. Motivation

Underwater acoustics is the study of the propagation of sound in water. Continuous monitoring of underwater sounds has important applications in diverse fields such as ocean wildlife and the engineering of renewable energy systems based on wave and tidal power. The resulting information can be used to monitor phenomena such as underwater animal presence, the impact of shipping and other traffic and the effects of equipment such as wave energy converters [1, 2] on underwater sound levels [16]. The resulting data can be used as input to simulations which are used to make good engineering and policy decisions about the likely impact of renewable energy systems in the ocean.

Figure 1. A buoy during calm conditions - note solar panels and communications antennae

This paper describes the on-shore base station used to collect data from an off-shore underwater acoustic monitoring system. The system is used to collect underwater acoustic information in real time for a protracted period. The raw data is collected using hydrophones, which are underwater microphones. The hydrophones are attached to buoys which float in the ocean up to several kilometers from the coast, and continuously monitor underwater sound. Raw data from the hydrophones is sent by wireless connection to the base station which is on shore. The base station receives the data, compresses it, and sends it on to the data processing centre where it is stored or used for real-time analysis or simulation. A single base station may receive data from multiple buoys on the ocean, therefore the base station must be

capable of receipt and compression of very large amounts of raw hydrophone data in real time.

Both the buoys and the base station are based in a remote location, which makes on-site maintenance time-consuming and costly. Furthermore, if the system fails, the opportunity to collect data in real-time will be lost, and there will be gaps in the record of sensor data. Therefore, the system is designed to be highly reliable, and to work autonomously for months or years at a time without human intervention.

The main contributions of this paper are:

- We describe our experience of engineering a base station for real-time processing of ocean hydrophone data.
- We provide a multi-threaded model of parallel receipt, compression and retransmission of sensor data in real time.
- We describe a novel multithreaded mechanism for safely queuing data between threads.
- We experimentally evaluate the scalability of our system, and show that a single base station can scale to receive data from many remote buoys in real time.
- We provide preliminary results on compression of acoustic data in real time.

The rest of this paper is organized as follows. Section II provides an overview of the system used to collect data for analysis and simulations. In Section III the off-shore hydrophone and data capture system is described. Section IV provides details of our on-shore base station and in particular discusses measures to improve reliability and real-time data capture. In Section V we present an experimental evaluation of our data capture system, and investigate scalability and data compression. Finally, Section VI describes applications of our system, and discusses plans for future work.

II. System Overview

The capture of signals from sensors located in real world environments, as seen in figure 1, necessitates the deployment of additional hardware(base station) to bridge the gap between the sensor and back haul network and also possibly to aggregate the sensor data into a more consumable

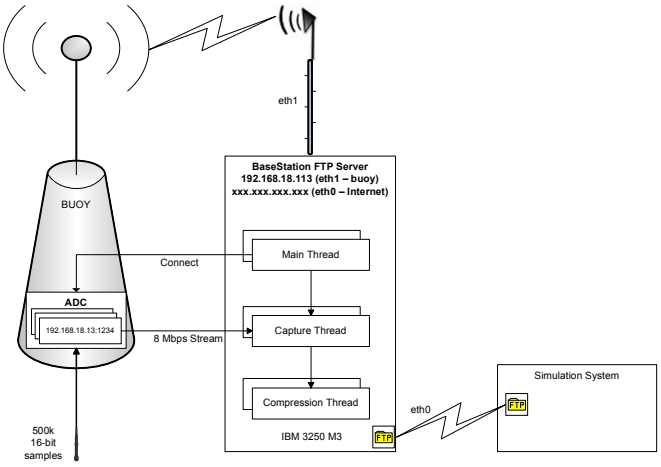

Figure 2. Overall System Architecture

format. It is therefore likely that this base station will also reside in a remote and perhaps inhospitable location. Hence, the administration of this machine may prove to be difficult and at times impossible. In order to ensure continuous, real-time capture of the sensor signals it is therefore imperative that the base station machine be as self-sufficient as possible, it must "stay up". However, it must also be sufficiently streamlined to be able to "keep up" with the stream of signals originating from the sensors and scalable enough to handle the real-time signal streams from several sensors in parallel.

In broad terms, there are three main components to the architecture of the system as seen in figure 2. First, the data acquisition system which in our case is the offshore hardware which consists of an underwater sound recorder called the hydrophone and an analog to digital converter to transmit the values recorded by the hydrophone. This is explained in further detail in section III. The second important component of the system is the base station which is an IBM server residing in some remote location. The server is responsible for the receipt and compression of the data from the analog to digital converter. Since the hydrophone produces data at a high rate (8Mbps), the server employs a three threaded model to keep up with the data influx which is further explained in section IV-D. The data is then stored

on an FTP file store with a very large storage capacity which further sends the compressed file onto an information system or simulation system of choice. These systems either try to simulate the conditions underwater or predict something based on real time data. Therefore the goal of the overall system is to provide a reliable and scalable way to acquire and send data to these analysis engines.

III. Offshore Hardware - Analog to Digital Converter

To ascertain the acoustic signature of an offshore location, a hydrophone is used to record the raw acoustic signals. These signals can be sampled at various rates, but for this application, the highest sampling rate was selected, which is 500,000 samples per second of 16-bit depth. These raw voltage signals will be converted to a digital stream of 16-bit integers by an analogue to digital converter (ADC) housed on a buoy. This system takes as input the raw hydrophone signals and outputs a converted stream of integers over an ethernet port. In TCP terms, the resulting packet stream is 8Mbps and a consuming application (on the base station), opens a TCP socket connection to a well known port on the ADC. Each connected hydrophone will have a corresponding and unique port number, so multiple hydrophone streams can be captured in real-time and in parallel.

Ethernet and the TCP protocol in particular were chosen primarily for its robust nature and prevalence, with very good software tools available both for monitoring and software development. This lends itself well to deployment in inhospitable environments, and in this case the use of wireless data transmission, with the base station software written to be able to react to communications interruptions.

A. Data Format

The structure of the header is as follows:
magic number - 64 bits - always $0xc0c0c0c0c0c0c0c0$
hydrophone id - 64 bits - unique id for each hydrophone
version number - 64 bits - software version number
timestamp - 64 bits - current ADC time stamp
sampling rate - 64 bits - current sampling rate

One issue with the data stream generated by the ADC is that the 16-bit integers are in big-endian format whereas commodity x86 hardware (base station) is a little-endian architecture. This places additional processing requirements on the ADC to convert these integers to little-endian prior to insertion into the TCP stream. Additionally, it may be necessary for a consuming application/simulations system to be able to analyze the signals in the time-domain (as compared to signal processing in the frequency domain). Therefore it is important that the ADC's clock time be also transported in the headers within the data stream. To this end, a data header is added to the stream every second, and is prefixed by a magic number: $0xc0c0c0c0c0c0c0c0$. This number is a marker to the analysis or simulation system that date/time/sampling rate information is available which can then be used to modify the simulation.

IV. Base Station

The base station software scaling is one process per known sensor. But within each process, there are a number of challenges in keeping up with the real-time capture of the data stream. Due to the harsh nature of the environment, it is expected that communications between the buoy and the base station will be sporadic. When communications have been established, signal capture commences and will continue while the connection persists. The captured signals must then be presented to a consuming application or simulation system, which will likely be in a remote location, possibly in a data center. As such, just streaming bytes directly through a standard wired back-haul network connection to the remote processing/simulation location may prove costly, which can be mitigated by having a compressed signal stream, as well as requiring constantly available reception hardware at the remote site. A better solution would be to present the signal data in a way that a remote application can pull the signal data when it wants, ideally through FTP. The use of FTP however requires files to transport not at real-time. So to support this, firstly the signal stream must be converted into a series of files. Again, the same

constraints apply, i.e. file creation/writing must keep up with the real-time stream and must be robust.

This means there are four distinct operations that the base station process must perform. Firstly, it must monitor the network connection to the sensor/ADC. Secondly, it must capture the signal stream. Thirdly, the stream must be written to disk and stored in a series of files. And finally, these files must be compressed. The base station must also provide a single standalone FTP server that can host access to these files.

A. Reliability

As the base station machine is expected to be available for capturing signals for an extended period of time, several important factors must be taken into account to achieve this up time. The physical hardware to be used is an important consideration: a commodity laptop would be less suitable than an IBM x3250 server in a lockable cabinet. An equally important decision is the selection of an operating system (OS) to run on this hardware. A minimal OS distribution will have fewer software components that are prone to failure. In particular there is no requirement at all for a windowing system, which greatly reduces the complexity of the system software. Ultimately, a stable OS distribution with a modern networking stack is the most suitable. For this reason, Ubuntu server edition was chosen, and two versions, 10.04 LTS and 12.04 LTS were evaluated. Due to the much improved networking backed by the later kernel (3.2), Ubuntu 12.04 LTS was chosen.

Just as important, is the architecture of the capturing software. A multi-threaded approach, with a dedicated thread spawned to handle each stream for individual sensors was investigated. In theory, this will work, but the overhead in managing many threads when the number of sensors increases starts to become a factor in ensuring application stability. If one sensor stream causes problems in one thread it may affect all other streams in all other threads. The base station is designed to be deployed in a remote location, which makes restarting the data capture application more difficult, which could result in signal loss for hours or days. For this reason, a single sensor stream per process approach was selected. This removes the possibility of stream cross-over effects and ensures a more robust application architecture.

B. Scalability

The software architecture for the base station was designed with reliability in mind, but scalability is also an important design issue. As with reliability, there can be significant disadvantages to servicing all sensors using multiple threads within a single OS process. Within a single process, multiple threads interact through shared variables and are synchronized with primitives such as locks. However, the interactions of threads and synchronization can result in unexpected performance behaviour where some threads can become starved of work or CPU time. This is particularly true when threads are shared among different streams, for example where a single thread might do data compression for all streams. Furthermore, within a single process multiple threads share resources, particularly data structures for memory management. As the number of threads rises, contention for shared resources such as memory management can result in poor scaling.

The balance between many threads per process and more processes with less threads is often difficult to arrive at. With more threads per process there is potentially a performance boost if the threads share data. With more processes there is better reliability, with problems in one thread not having the potential to spill over to another.

With one process per sensor approach, a single application instance is responsible for capturing the sensor signal, storing it and compressing it. This scales to multiple sensors in two different ways. Firstly, additional application instances (processes) can be executed, with each process dedicated to a single sensor signal stream. In this way, a single base station machine can handle multiple sensor streams in parallel. When the processing capabilities of the hardware, in terms of keeping up with the signal rate, have been exhausted, additional base station machines can be added. Each additional base station machine can serve a different set of sensors.

C. Queuing

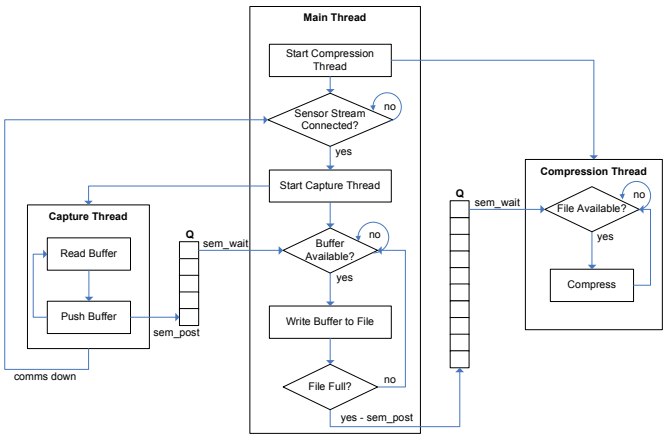

Figure 3. Interaction of different threads in the basestation

At the heart of this thread interaction is a producer-consumer style queuing system. Due to the critical positioning of this, it is imperative that it does not impose any significant run time overhead on the application and that it is thread-safe. There are two main features of the queue that reduce the run time overhead. Firstly, it is thread-safe, but lock-less, meaning that no unnecessary system calls are required to synchronize access to the queue elements. Secondly, element insertion and retrieval are zero-copy operations, removing the need to invoke an expensive memory copy.

The queue allocates a ring-buffer and is built by using two semaphores to control access, a "push" semaphore to control adding elements to the queue and a "pop" semaphore to control removing elements. The "push" semaphore is initialized to the number of available spaces in the queue, e.g. 16. The "pop" semaphore will be initialized to zero, i.e. there is currently nothing in the queue. Two additional variables "push_at" and "pop_at" refer to the exact position in the ring-buffer for pushing and popping elements.

When a producer wants to add to the queue, it first checks for available space by calling sem_wait on the "push" semaphore. This blocks until there is space available. It then uses the "push_at" queue space, increments "push_at" and calls sem_post on the "pop" semaphore to alert consumers that there is a new item in the queue.

```
struct queue {
sem_t push;
sem_t pop;
int push_at;
int pop_at;
int length;
size_t elem_size;
void *data;}
```

This decrements the number of available spaces in the queue. When a consumer wants to remove something from the queue, it checks for an entry in the queue by calling sem_wait on the "pop" semaphore, which will block if there are no elements, pulls the queue item as referenced by "pop_at" and calls sem_post on the "push" semaphore to notify producers that there is an extra available space in the queue.

D. Thread Interactions

As described in Figure 3 there are two distinct thread interactions. The main thread is a consumer of buffers produced by the data capture thread and also a producer of file descriptors for use by the compression thread. It establishes communication with the buoy's ADC by using standard TCP sockets. When it detects that the communications link is up, it spawns a capture thread to handle this communication stream and capture the signals. It will then use a capture queue to wait for buffers to be produced by the capture thread. The capture thread receives bytes from the socket and stores them in a temporary buffer. When this buffer is full, it posts the buffer to the capture queue and continues to receive data from the socket. The main thread will be notified of the new buffer and will write this buffer to a file. When the file has reached a pre-determined maximum size, it will be flushed and its file descriptor added to the compression queue. The compression thread will in turn be notified and will start to compress the file by shelling out to invoke a compression script. This multi-producer-consumer model enables the base station process to keep up with the real-time sensor signal stream whilst in parallel preparing a compressed file of sensor signals for download by a remote application/simulations system.

V. Experimental Results

A. Scalability

In order to understand the scaling characteristics of the capture application it is necessary to create a simulation environment which can mimic the traffic generated by multiple hydrophones. This environment is an extra application running on independent hardware that generates 8Mbps of data. Throttling the traffic to 8Mbps is achieved by surrounding the network calls by gettimeofday calls and sleeping by the number of microseconds in which they differ.

A number of instances of this hydrophone simulation application and the capture application are executed in parallel, to help ascertain the scalability of the capture application. From the graph in Figure 4, as the number of parallel capture processes increases, the amount of time spent in userspace/system calls and the available CPU idle time start to converge.

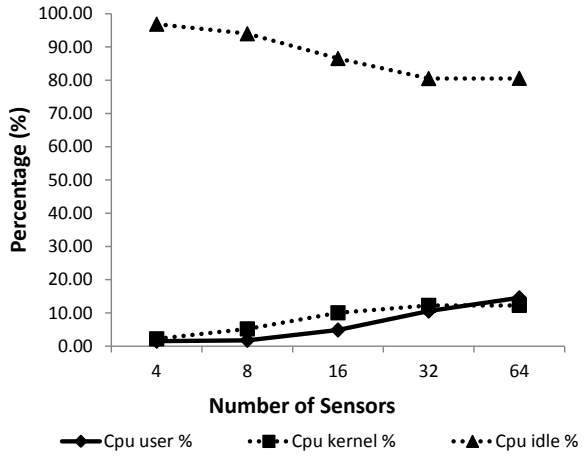

Figure 4. Scaling characteristics of the application

The graph in figure 4 shows that a significant amount of time is spent in the Linux kernel, in fact the system profiler reports that 56% of the per–process run time for the capture process is spent in the select system call. This is consistent with the software architecture which yields CPU time in an operating system friendly manner. There is no busy waiting within the application, with all blocking performed via system calls, namely select for incoming network packets and sem_wait for incoming messages from other threads.

For this simulation, the capture application ran on a quad core, hyper threaded Lenovo D20 Intel(R) Xeon(R) CPU (E5640) @2.67GHz with 6GB of RAM. The operating system was Ubuntu Server 12.04 64-bit with a 3.2.0-24-generic Linux kernel. The traffic generator ran on a Lenovo S10 Intel(R) Core(TM)2 Quad CPU (Q6700) @2.66GHz with 2GB of RAM. The operating system was Ubuntu Server 12.04 64-bit with a 3.2.0-23-generic Linux kernel.

B. Compression Statistics

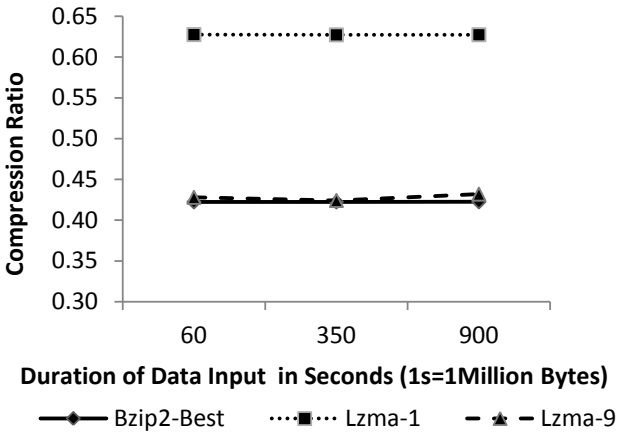

Figure 5. Comparison of Compression Ratios across the 3 compression methods

In keeping with the theme of "Staying up and keeping up is critical" as discussed in section 2, some tests were performed to see how well the compression thread keeps up. This section describes these compression statistics. The objective of performing these tests was two–fold. One was to find out which compression algorithm would be best suited for the output data from the hydrophones and in choosing this algorithm, will the compression thread fail to keep up at any point in time including the worst case input?

The output from a hydrophone as described in the previous sections, is the measure of the acoustic intensity in the area around the hydrophone. It is expected that for significant periods of time (calm waters) that there will not be significant variation in the signal. With this in mind, three compression schemes were chosen, Bzip2-best[8], Lzma-1 and Lzma-9[9]. Three tests(60s,

350s and 900s) were performed on each of these compression schemes with different transmission time(which effectively means different filesizes to compress). Each second of data transmission corresponds to 500,000 samples of 2 bytes each. This implies each second of data transmission corresponds to 8Mbps(Mega bits per second).

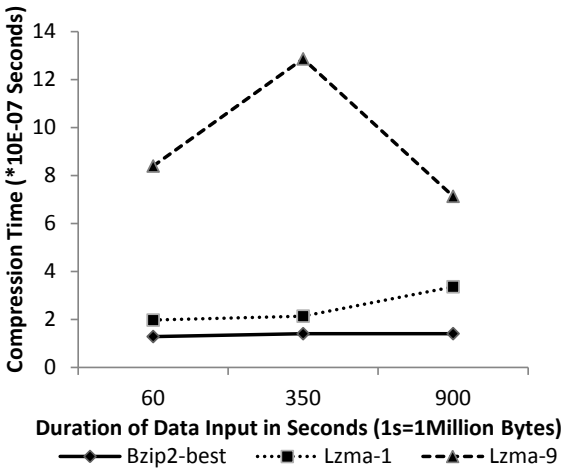

Figure 6. Comparison of Compression Time per byte across the 3 compression methods

Figure 5 shows the compression ratio offered by the three compression schemes for the different inputs given to them and its very evident that lzma-9 performs worse than lzma-1 and Bzip2-best. It can also be observed that the compression ratios offered by lzma-9 and Bzip2-best are about the same. But the difference arises in the time required to complete the same workload. Figure 6 shows how much time each algorithm takes to compress one byte, for three different workloads. The scale of the y axis in this figure is in the order of 10^{-7}. As is evident, Bzip2-best performs the best in these test conditions which are very close to the deployment environment.

For this simulation, the compression thread (all three compression schemes) ran on a quad core, hyper threaded Lenovo D20 Intel(R) Xeon(R) CPU (E5640) @2.67GHz with 6GB RAM.

VI. Applications and Future Work

Hydrophones can operate over a wide range of frequencies, both low and high. Shipping is an example of low frequency underwater noise and with the raw acoustic signals available in real-time a shipping simulation application could predict the course of various ships overlaying this with weather information to predict course corrections. Similarly, dolphin echolocation clicks are an example of high frequency noise and a dolphin tracking application could use the real-time signals to identify the presence of dolphins nearby and attempt to estimate the population for the locality. For the deployment of wave energy converters, we can also quantify the converters add a significant amount of acoustic noise to the sites underwater acoustic signature.

At present the base station software is a packet listener only. In other words, it establishes a connection to the ADC and then just receives packets, there is no protocol involved for "chatting". An interesting addition would be to add such a capability, allowing for a command and control style communications with the ADC. It could be instructed to change the sampling rate or to power down wireless communications for "x" hours if it is winter time. Furthermore, the current compression algorithms (bzip2, lzma) could be replaced with a more suitable high frequency audio compression algorithm. As the range of frequencies is quite large (˜500kHz), standard mp3 techniques are insufficient as consuming applications may require a loss-less algorithm for more accurate species/engine noise identification.

VII. Conclusion

In this paper we have described the design and implementation for an on-shore base station that collects very large amounts of underwater acoustic data from off-shore buoys. The data is collected in real time, and can be used for analysis and simulation of underwater ocean environments, and in particular for monitoring underwater animal activity, and understanding the effects of bringing renewable energy technologies such as wave energy capture systems into the environment.

Our base station system is designed to receive and process large amounts of sensor data in real time, and a single base station can receive data from many buoys in the ocean. Our scalability

tests show that our system easily scales to handle 64 separate streams of sensor data. We capture, compress, and retransmit data in separate threads ensuring high performance capture and compression that reaches real-time requirements. Finally we present experimental measurements of data compression rates on sensor data which suggests that high levels of compression are possible, at least during the majority of time when there is little more than general background noise.

Our base station will soon be deployed on the west coast of Ireland, where it will be used to collect raw underwater acoustic data in the Atlantic Ocean in real time over the course of a full year at a rate of 500,000 16-bit samples per second. This is equivalent to 8Mb/s or around 28.8TB of data over the course of a year. By sampling at a high rate, i.e. 500,000 samples per second we give marine biologists or analytic engines a very fine grained input which may ultimately result in more accurate population counts for specific marine species.

Acknowledgement

This work was supported by the Sustainable Energy Authority of Ireland (SEAI) and the Marine Institute of Ireland. Aravind Vasudevan's research is supported by the IRCSET Enterprise Partnership Scheme in collaboration with IBM Research Dublin, Ireland. David Gregg's research is supported in part by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).

References

[1] Nolan et al., "Design and Control Considerations for a Wave Energy Converter", Irish Signals and Systems Conference ISSC 2004, Belfast. pp. 475–480. ISSN 0537–9989.

[2] Aaron Zettler-Mann, "The Effects of Wave Energy Converters on a Monochromatic Wave Climate", Honors Thesis at University of Colorado Boulder.

[3] Stevens et al., "UNIX Network Programming, Volume 1: Networking APIs: Sockets and XTI", 2nd Ed., Prentice–Hall, Inc., 1998.

[4] Provos et al., "Scalable Network I/O in Linux", USENIX Annual 2000 Technical Conference.

[5] Metz, "Protocol Independence Using the Sockets API", USENIX Annual 2000 Technical Conference.

[6] Rago, "UNIX System V network programming", Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[7] J. Seward. `bzip2`, version 1.0.5 http://www.bzip.org/1.0.5/bzip2.txt

[8] `Lzma`, version 4.32.7 http://tukaani.org/lzma/

[9] Bryant et al., "Scaling linux to the extreme", In Proceedings of the Linux Symposium 2004, pages 133–148.

[10] Kleen, "Linux multi-core scalability", In Proceedings of Linux Kongress , October 2009.

[11] Yan et al., "OSMark: A benchmark suite for understanding parallel scalability of operating systems on large scale multi-cores", 2nd International Conference on Computer Science and Information Technology, pages 313–317, 2009.

[12] Yan et al., "Scaling OLTP applications on commodity multi-core platforms", IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS), pages 134–143, 2010.

[13] Kolar et al., "Complex Real-Time Environmental Monitoring of the Hudson River and Estuary System." IBM Journal of Research and Development, vol. 53, no. 3, Paper 4, 2009.

[14] Kolar et al., "Stream analytical processing of acoustic signals for cetacean studies and environmental monitoring of ocean energy conversion devices", Proceedings Oceans 11 – IEEE–OES Santander 2011

[15] Richardson et al., "Marine Mammals and Noise", San Diego, CA: Academic Press, 1995.

[16] "Conservation Plan for Irish Cetaceans", Department of Environment, Heritage and Local Government, Public Consultation Draft, October 2009.