PC CLUSTERS FOR SIGNAL PROCESSING: AN EARLY PROTOTYPE

John Chapin

MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
jchapin@lcs.mit.edu

Andrew Chiu and Roger Hu

Vanu, Inc.
15 Ward Street
Somerville, MA 02145
agchiu@vanu.com, rhu@vanu.com

1. ABSTRACT

A cluster of commodity personal computers (PCs) is an attractive platform for signal processing, especially for applications like sensor arrays where there is significant parallelism in the computation. This paper reports the results of experiments with several small cluster configurations to explore the opportunities and limitations of this approach. Experiments included uniprocessor and multiprocessor PCs, 100 MBit/sec and 1 Gbit/sec ethernet. All experiments were receive-only, starting from raw spectrum samples provided by a wideband A/D card, and all signal processing code ran as application processes on top of standard Linux. The applications used to drive the experiments were delay and sum beamforming and the simultaneous reception of multiple AMPS cellular channels. The currently standard PCI I/O bus (32 bits at 33 MHz) was the most significant bottleneck in the experiments. This suggests that the arrival of the next generation of commodity PCs, with faster I/O busses (such as 64-bit PCI at 66 MHz), may open the door to their use as a low-cost platform for a variety of low-end realtime signal processing applications.

2. INTRODUCTION

Commodity PCs continue to improve rapidly in processor speed and memory size while maintaining outstanding price/performance. Several years ago it became possible to implement an FM radio receiver in software on a general-purpose PC, with all processing stages in software except for initial amplification and downconversion to baseband [1]. Subsequent work has resulted in software implementations of a variety of RF protocols. The commodity marketplace has also seen the recent introduction of low-cost products with significant bandwidth improvements compared to previous PC standards: RAMBUS memory systems (2x memory bandwidth), 64-bit 66-MHz PCI buses (4x I/O bandwidth), and gigabit ethernet over standard copper wiring (10x networking bandwidth).

We believe that the commodity market is about to reach the point where clusters of standard PCs are a compelling platform for many large-scale signal processing applications. Applications for which clusters of PCs will be compelling in the near future are those with the following properties. The computation must be parallelizable or pipelineable with pairwise connection bandwidth less than 1000 Mbit/sec (bisection bandwidth can be significantly higher). The application cannot require small size, low power, or physical ruggedness from its computational platform. The application must tolerate multi-millisecond latency from input signal to output signal or response action. For applications with these characteristics, which include many sensor array processing problems, the development time and deployment cost for implementations using PC clusters are likely to be far less than for implementations using traditional DSP approaches.

There are two major challenges in working with PC clusters. First, when compared to dedicated DSP systems, commodity PCs are starved for memory bandwidth and I/O bandwidth, a situation exacerbated by a cache and memory system design not tuned for data streaming. Second, individual PCs are not reliable enough for most signal processing applications, so the cluster must be designed to move computation to a backup machine when a failure occurs. We have begun a research project to characterize the limitations and opportunities of PC clusters more precisely, and to develop a software system with the features needed to support high-performance, reliable signal processing applications.

This paper describes experiments on the first prototypes built in this project. We describe the questions that motivated the experiments (section 3), the prototypes (section 4), present latency and jitter measurements (section 5), performance measurements (section 6), and conclude with lessons that shape further work in this area (section 7).

3. QUESTIONS

This early prototype experiment was designed to answer the following questions relevant to all uses of PC clusters as signal processing platforms. In this discussion, the term "sample-rate" refers to the high data rate characteristic of DSP sample streams, anywhere between 20 and 1000 Mbit per second.

Can TCP be used as the networking protocol?

TCP is the standard reliable stream protocol used in internet applications [2]. It is attractive for DSP cluster applications for several reasons. It is provided with every operating system, and vendors spend considerable effort on improving TCP performance. Commodity network interface cards provide on-board support for TCP, reducing host processor CPU load [3]. Because TCP provides end-to-end reliability, it enables the system designer to choose commodity networking components that occasionally drop packets.

However, there are potential problems with TCP because it was initially developed for wide-area networks with lower bandwidths, higher delays, and greater packet loss rates than the interconnect used in a cluster. Issues to investigate include the CPU workload induced by generating and receiving sample-rate streams, the jitter induced by TCP retransmissions and receiver window management, and the latency induced by the buffer sizes required to achieve sample-rate throughput. If any of these attributes are unacceptable, a networking protocol tuned for the characteristics of clusters will need to be adopted. There is an emerging standard in this area, M-VIA, but TCP is substantially more mature.

What is the latency and jitter added to a data stream when it passes through a PC?

The limited processing power of individual PCs means that the system is likely to contain pipelines of multiple PCs. Each PC in the pipeline will add additional latency and jitter due to networking and OS characteristics, irrespective of the actual computation being performed. Knowing these overhead values will enable predicting whether specific applications, with known latency and jitter tolerance and known computational requirements, can be implemented on PC clusters or not.

4. PROTOTYPE

The hardware components used in the various experiments are detailed in table 1.

All application software runs as normal Linux processes at user level, with the exception of the Gage driver whose only function is to transfer data from the Gage card to user-level buffers.

The application software is structured as a library of C++ objects, each of which performs a dedicated function such as filtering or demodulation. The objects are instantiated at run time and connected together into data-processing pipelines, in which data is pulled by the sinks rather than pushed by the sources. See [1] for a description of an earlier system that inspired the current software architecture.

The library contains objects which application writers can use to splice together pipelines on separate machines into a single distributed pipeline. Data is sent over the network as a TCP stream. In the 100 MBit/sec network, TCP achieves 90 MBit/sec with 4 kbyte buffers when the machines are otherwise idle. In the 1 GBit/sec network, TCP achieves 407 MBit/sec with 64 kbyte buffers.

The current prototype does not tolerate failure of a machine.

5. LATENCY AND JITTER

Latency and jitter were measured in the following way. The two XPS machines were connected by 100 MBit ethernet, with machine A generating a continuous stream of incrementing integers and machine B reflecting the stream back to A. Machine A noted the timestamps at which particular values in the datastream were sent and received (timestamps on these machines are accurate to less than 5 microsec). Subsequently, a third machine was added to the loop, so the data was sent from A, to B, to C, and then returned to A. 50,000 samples were collected; the data stream flowed at 70 Mbit/sec.

In the two-machine case, average latency was 4.3 msec and jitter (stddev of the latency distribution) was 0.434 msec. In the three-machine case, average latency was 6.5 msec and jitter was 0.625 msec. Therefore the addition of machine C added 2.2 msec of latency and $\sqrt{0.625^2 - 0.434^2} = 0.45$ msec of jitter.

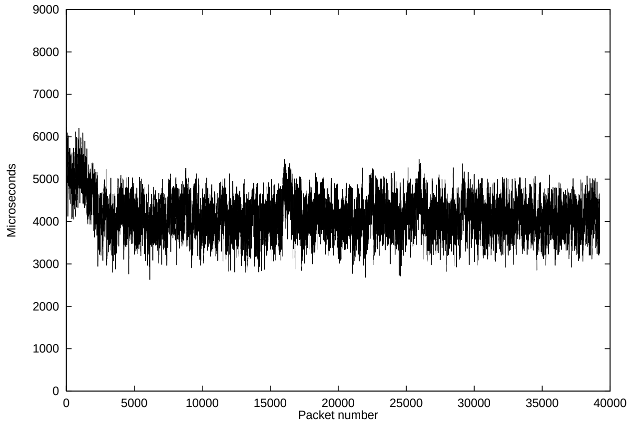

Figure 1: Latency per packet for 2-machine loop.

Figures 1 and 2 plot the latency per packet for the two-machine and three-machine cases. It is interesting to observe the sawtooth pattern in figure 2 (which is even more

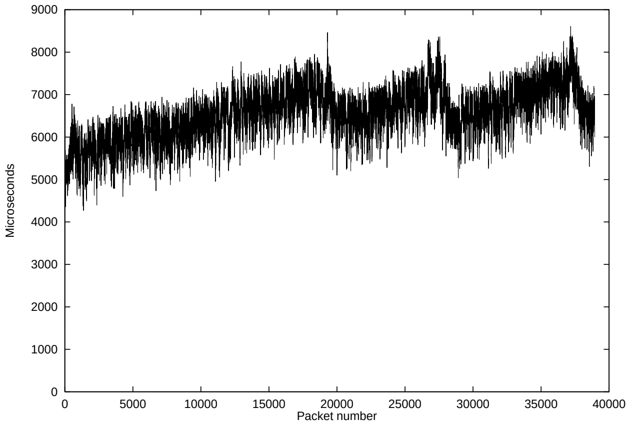

Figure 2: Latency per packet for 3-machine loop.

pronounced in a four-machine loop test, not shown here). We hypothesize this pattern is due to TCP rate adaptation mechanisms [2] and plan to investigate further.

The measured jitter values are standard deviations, not maxima. Figure 2 shows that occasional packets in the three-machine test have 8 msec latency. We reran the experiments in OS maintenance mode where no other processes are allowed to run on the machines; there was no significant difference. This fact, plus the observation that the high latencies occur irregularly at the peak of the sawtooth pattern, suggests that the high latency is due to TCP retransmission after packet loss.

6. PERFORMANCE MEASUREMENTS

6.1. Delay and sum beamforming

Figure 3 shows a small cluster of two machines that was configured to perform delay and sum beamforming for one output from four antenna inputs. Machine A computed a partial sum on its two antenna inputs and forwarded the sum to machine B. Machine B computed the final result from its two antenna inputs.

In this configuration, the performance is limited by machine B, since it performs all of the functionality of machine A with the added load of computing the final sum. When the antenna inputs supply 450 kSample/sec, with each sample represented as a 64 bit complex float, machine B is at 97% utilization and machine A is at 40% utilization.

When the antenna inputs are replaced with simulated sample sources, the cluster is able to process 1 Msample/sec. Thus, the use of the Gage A/D card adds considerable overhead to the system. During its operation, the Gage card causes a high number of processor interrupts to occur when it transfers samples from the A/D converter to system memory. These interrupts reduce the efficiency of the processor

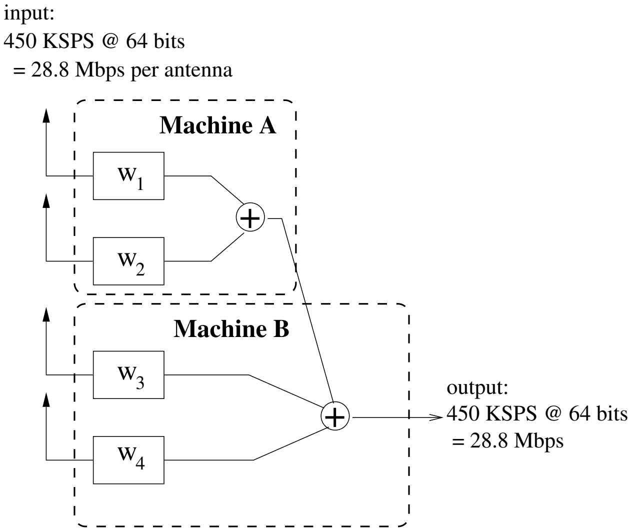

Figure 3: Cluster configuration: four antenna delay and sum beamforming.

by forcing the processor to pause its current computation to service the interrupt. We hypothesize this is the dominant cause of the reduced performance.

6.2. Multiuser beamforming

Figure 4 shows a multiuser beamforming antenna array with six antennas that was constructed with five machines and a gigabit ethernet switch. Two outputs representing two directions were computed. In this configuration, three machines (A, B, and C) each had two input antennas. These three machines computed partial sums for the two directions and sent each sum to two other machines (D and E). Machines D and E each took in three data streams (one from each of machines A, B, and C) and computed the final sum.

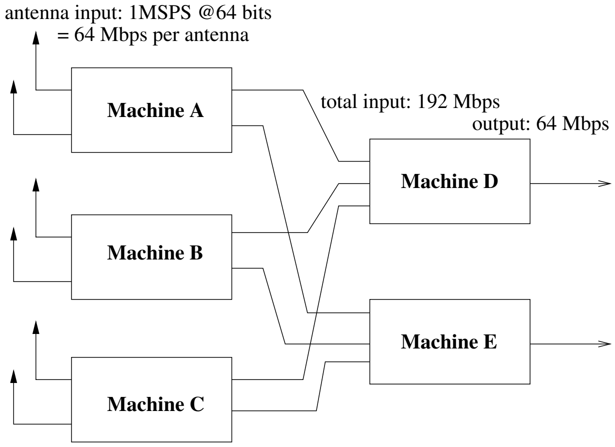

Figure 4: Cluster configuration: multiuser beamforming.

Dell Dimension XPS	4	400 MHz Pentium II or 450 MHz Pentium III 128 MB RAM, 512 Kb cache PCI I/O bus (32 bits @ 33 MHz) 3Com 3C905B 100 Mbit/sec ethernet interface NetGear GA620 1 Gbit/sec ethernet interface Linux 2.2
Dell Dimension XPS r	2	800 MHz Pentium III 128 MB RAMBUS RAM, 256 Kb cache PCI I/O bus (32 bits @ 33 MHz) 3Com 3C905B 100 Mbit/sec ethernet interface 3Com 3C985B 1 Gbit/sec ethernet interface Linux 2.2
Dell 6300	1	four 500 MHz Pentium III, each 512 Kb cache 784 MB RAM PCI I/O bus (32 bits @ 33 MHz) 3Com 3C905B 100 Mbit/sec ethernet interface 3Com 3C985B 1 Gbit/sec ethernet interface Linux 2.2
Netgear FS524 switched ethernet hub	1	1.2 gbit/sec backplane 148,000 packets/sec (64 byte packets) 60 microsec max latency
Extreme Summit1 gigabit switch	1	17.5 Gbps non-blocking switch fabric 11.9 M packets per second
Gage CompuScope 1250 A/D	2	10 MHz bandwidth 20 Msamples/sec 12 bits/sample (padded to 16 bit data value) installed in the PCI bus receives baseband signal from custom downconversion board

Table 1: Hardware components used in experiments.

This configuration was tested by using simulated sources for each of the antennas. The maximum data rate that this cluster could support was 1 Msamples/sec, with each sample represented as a 64 bit complex float. Table 2 shows the CPU utilizations of the five machines when operating at this data rate.

Machine	CPU load
A (Pentium II 400)	90%
B (Pentium III 800)	16%
C (Pentium III 800)	24%
D (Pentium III 500)	50-90%
E (Pentium III 450)	45-90%

Table 2: CPU utilization for multiuser beamforming.

The limiting factor in this application was the memory system. The TCP protocol makes a copy of every packet that is sent. When we attempt to increase the input and processing data rate on machine A, the output data rate tops out at about 128 Mbps (two streams, each 64 Mbps). The CPU load also tops out at 90%. If the machine was processor or I/O limited, the load would have increased to 100%. However, when the machine is memory limited, the CPU stalls while it attempts to access the memory bus.

6.3. AMPS multichannel reception

A cluster of 450 MHz machines connected by 100 MBit/sec ethernet was configured to perform channel selection and decoding in the AMPS cellular band from 870 MHz to 880 MHz. Each of the four machines performed a different function (see figure 5). Samples from the antenna were collected on the band selection machine, which extracted two 200 kHz sections of the 10 MHz cellular band (machine A). Each of these 200 kHz sections was sent to a channel selection machine (machines B and C). Each channel selection machine extracted six 30 kHz AMPS channels from its input section and sent those channels to the demodulator (machine D). This machine performed FM demodulation and audio filtering on each of the voice channels and saved each audio stream to disk. In total, twelve voice channels were simultaneously produced on the demodulation machine. Connecting each voice output channel in turn to a speaker confirms that the system produces comprehensible voice output for all channels.

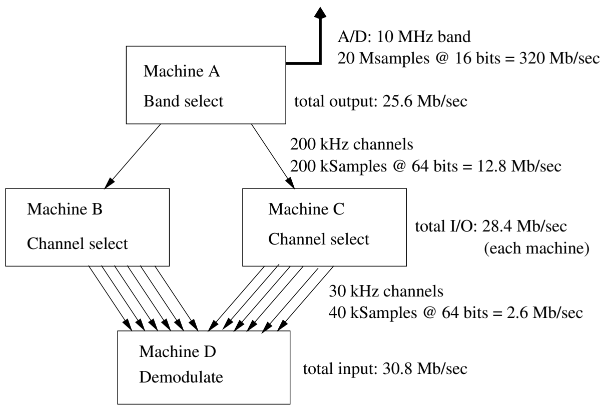

Figure 5: Cluster configuration: multichannel AMPS processing.

The system's capability is limited by the band selection machine, and within that machine either by the PCI bus or by memory bandwidth, not by processor capability. The following configuration was used to verify this fact. A machine takes, as its input, a 20 Msample/sec sample stream representing 10 Mhz of spectrum. A FIR filter with 256 taps is used to select a 200 kHz band and produce a 200 kSample/sec complex sample stream. This operation requires 51.2 million filter operations per second ($256 \cdot 200000$), and when this output data is discarded, the CPU utilization is at 60%. Adding network transmission of the output data increases the processor load to 65%. When the output bandwidth is increased to 400 kHz and the number of taps reduced to 128 (thus keeping the computational load constant at 51.2 million filter operations per second), CPU utilization jumps to 100% and the system begins to drop data. The most likely limitation is the PCI bus, since at 200 kHz sections it is being driven at about 50% of its maximum throughput by two competing streams, incurring repeated turnaround and arbitration penalties. Future experiments with a higher-speed I/O bus will enable testing this hypothesis.

Each channel select machine executes six channel selection filters on its input section, producing six 30 kHz streams (oversampled at 40 kSample/sec). Each channel selection filter is implemented as a polyphase FIR filter that performs frequency translation to baseband and bandwidth reduction [1]. This work plus the networking computation required to receive and transmit the sample streams only consumes 41% of the processor. If the band select machine were able to deliver a wider band, each channel select machine would be able to select at least 12 channels.

The demodulation machine executes twelve processing pipelines, each of which performs FM demodulation and filtering (using an FIR filter) and saves each audio stream to disk. Its total processor utilization for all 12 channels, including networking, is 50%.

When listening to an output channel on the speaker, there is a small but noticeable delay compared to a mobile handset receiving the same signal. We are investigating the source and exact amount of the cluster's latency more closely.

7. CONCLUSION

The experiments described here verify the intuition that PC clusters are currently viable for low-bandwidth signal processing tasks, up to around 10 MHz for simple tasks like band selection. The primary limitation is the I/O bus. The secondary limitation appears to be the interrupt load placed on the processor when too many small I/O transactions occur, but we have not verified this observation yet.

The former limitation will be addressed in the course of time as the commercial marketplace drives PCs towards ever-higher performance. If the latter limitation is confirmed, targeted development efforts may be required to support high-bandwidth signal processing, such as wideband A/D and D/A cards designed to minimize processor interrupts.

We are encouraged that the measured latency and jitter are low enough to support a range of important signal processing tasks, even without using an operating system tuned for tight real-time behavior. The PC cluster approach to signal processing appears promising.

8. REFERENCES

[1] V. Bose, M. Ismert, M. Welborn, and J. Guttag. "Virtual radios," *IEEE Journal on Selected Areas in Communications,* vol. 17, no. 4, pp. 591–602, April 1999.

[2] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols.* Addison-Wesley, 1994.

[3] `http:// www.alteon.com/ products/ acenic-data.html`

[4] H. J. Kim and H. S. Kim. "Cost-effective parallel processing for remote sensing applications," *1996 International Geoscience and Remote Sensing Symposium,* vol. 1, pp. 405–407. IEEE, 1996.