Real-Time Sonar Beamforming on a Unix Workstation using Process Networks and POSIX Pthreads

Gregory E. Allen

24 February 1998

Motivation

- Beamforming is a computationally intense algorithm which has traditionally been limited to expensive custom hardware. (GFLOPS)

- Current multi-processor RISC workstations post benchmarks that approach these capabilities at a fraction of the cost.

- Real-time performance has traditionally been beyond the capabilities of standard Unix.

- POSIX Real-time extensions (including Pthreads) with symmetric multiprocessing could allow a workstation implementation.

Objectives

- Utilize the process networks model, which is a natural way to describe signal processing systems.
- Implement high-performance, low-overhead process networks using lightweight threads.
- Implement and benchmark “digital interpolated beamformer” algorithms.
- Create a real-time beamformer on a workstation that can replace custom hardware.

Implementation

- Used Matlab to generate and test beam coefficients, and to verify beamformer output.
- Developed C++ code to implement process networks and digital interpolated beamforming.
  - Used existing Pthread and timer classes.
  - Developed a Process Network queue class.
  - Developed a sparse FIR filter (using coefficients from Matlab).
  - Developed other (more optimized) beamforming kernels.
  - Developed several different beamforming implementations for comparison.

Process Networks Queue Class

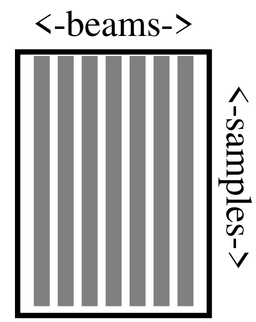
- Each process is a POSIX thread, and UNIX does the scheduling.
- Use Parks' rules for dynamic scheduling of process networks:
  - A processes blocks if it reads from an empty queue.
  - A process blocks if it writes to a full queue.
  - (We assume that queues are large enough to avoid artificial deadlock, and that the program is not unbounded.)
- Borrow the idea of a firing threshold (**Tp**) from Karp and Miller.
  - Queue threshold is max[Up, Wp, Tp].
- Intended to be optimized for DSP applications:
  - Processes operate directly from queue memory, so that no data has to be copied.
  - Data is guaranteed to be contiguous in memory to make up for the lack of circular addressing modes on a general-purpose processor.
  - There is a tradeoff between memory usage and performance.

Beamforming Implementations

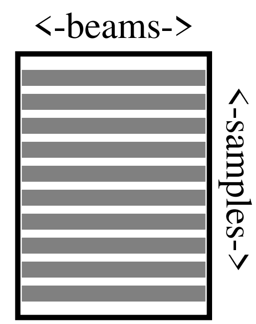
- Batch-mode, with a single thread of execution.
  - All sensor data is read into memory.
  - One task calculates results from memory to memory.
  - All results are written out to disk.
- Batch-mode, with multiple threads of execution.
  - Multiple threads divide the task of calculation.
  - How the task is divided can greatly affect the performance.
- Process Network, with a single thread beamformer.
  - A producer thread and a consumer thread provide an interface from memory to Process Network queues.
  - A beamforming thread takes samples from the producer, calculates the result, and sends it to the consumer.
- Process Network, with a multi-threaded beamformer.
  - Multiple threads divide the task of calculation.
  - "Thread pools" -- merging DSP and workstation algorithms.

Dividing The Beamforming Job

- Divide by beams
  - “Partial Beamforming”.
  - Low latency.
  - Low memory use.
  - Poor cache utilization.
  - A “DSP” approach



- Divide by samples
  - Good cache utilization.
  - Higher latency.
  - More memory usage.
  - A “workstation” approach.



- Tips for performance.
  - These are DSP algorithms, but they are running on a workstation.
  - Throughput is more importatant than memory usage or latency.
  - Keep each thread’s kernel calculations in the cache.

Results

- Used average time over 10 trials to calculate 10000 beam sets.
- All cases divided computation by samples, for best performance.
- Benchmarks performed on a Sun Ultra-2 workstation with dual 300 MHz UltraSPARC-II CPUs.
- 4 CPU benchmarks performed on a Sun SPARCserver 670MP workstation with quad 72 MHz HyperSPARC CPUs.

Beamformer type	Seconds 1 CPU	Seconds 2 CPUs	Speedup 2 CPUs	Speedup 4 CPUs
Batch-mode	0.532	0.270	97%	290%
Process network	0.682	0.360	90%	210%

Comments

- Kernel performance of about 19,000 samples per second per CPU.

- 33% overhead for process networks.

- Process network overhead is inflated because of data copying.

Expected Current Performance

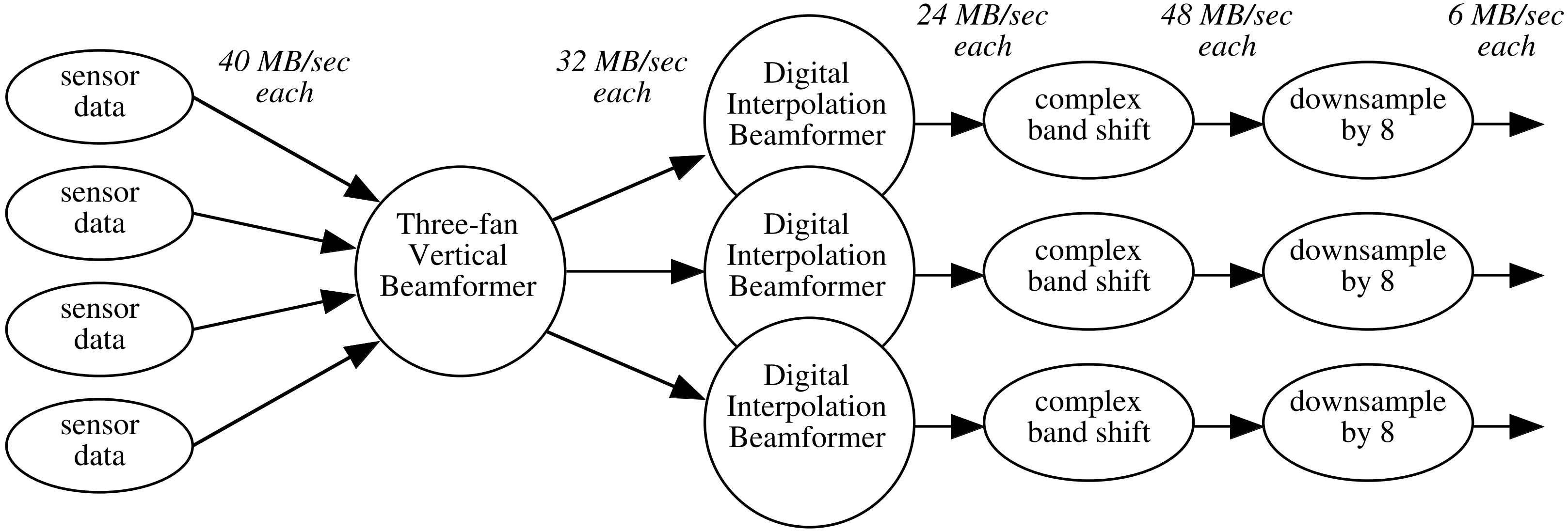
- The Process Networks queue has been rewritten for better memory-to-memory operation, which should substantially boost performance and reduce overhead. Previously, only consumers operated out of queue memory.
- Beamformer kernel (batch mode) routines have reached upwards of 35,000 samples per second per CPU (better than 80% speedup).
- Advanced Sonar Group (ASG) just installed a quad 300 MHz UltraSPARC-II workstation, and is ordering a workstation with 8 UltraSPARC-II CPUs running at 333 MHz.

Conclusion

- Realization of a real-time beamformer on a Unix workstation is on the horizon.
- A workstation beamformer could be implemented at a fraction of the cost of custom hardware.
- As workstations become even faster, custom hardware beamformers may become obsolete.

Future Work

- A three-fan vertical beamformer with 3 digital interpolation (horizontal) beamformers. Current estimate, 16 to 20 CPUs.



- An entire real-time sonar on a workstation?