Michael Maniscalco

email: michael@michael-maniscalco.com
github (compression and algorithm projects): github.com/michaelmaniscalco
github (c++ software architecture projects): github.com/buildingcpp
linkedin: www.linkedin.com/in/michael-a-maniscalco
website: www.michael-maniscalco.com

Veteran C++ developer, software architect and innovator with extensive experience in:

- low latency software development.
- greenfield architecture development.
- concurrency, multi-threading and lock-free/wait-free algorithms.
- generic programming and template meta programming.
- networking, kernel bypass etc.
- algorithm development, data compression, sorting, searching.
- authoring software for distributed architectures.

Portfolio:

github.com/michaelmaniscalco
github.com/buildingcpp

I maintain many github repositories which, I believe, are the strongest representation of my abilities as a software architect as well as as a C++ developer. I encourage readers to review these repositories in order to understand and gauge my skill set. Please see the "*Portfolio: github, algorithms, research & inventions*" section of this document for a more detailed description of some of these works.

Lime FinTech
Architect & Principal Developer
December 2020 - present

Primary Roles and Responsibilities:

Architected and implemented next generation low latency market data product:

- Highly deterministic sub-microsecond low latency performance.
- Responsible for all architectural decisions.
- Responsible for authoring the entire code base.
- Responsible for all documentation.
- Responsible for all early client interaction and support.
- Project produced a large collection of libraries, each designed to provide specific functionality such as kernel interaction (file descriptors, shared memory, asynchronous threading, etc), networking, basic framework for building distributed architecture, messaging, configuration, licensing, custom logging/instrumentation library, asynchronous shared memory/file backed storage, etc.

- **Highly asynchronous architecture with virtually no locking:**
  Employed my ‘Work Contract’ asynchronous library to make virtually all processing asynchronous. My work contract approach also facilitates lock free designs - which is heavily exploited in the products that I have built for Lime.

- **Restructured networking (kernel and ef_vi) library for reduced latency:**
  Updated my previously developed networking library to be even lower latency and coupled with my work_contract based asynchronous library.

- **New messaging library and protocol parsers (heavy template meta programming):**
  Extremely robust, trivially easy to use, messaging library using template meta programming, compile time hashing and message routing for ultra low latency message parsing and marshaling. TMP based pattern for easily defining new protocols and the messages of those protocols. Lock free multi-producer message streams.

- **Distributed architecture:**
  Created an entire suite of specialized services in a distributed architecture to manage low latency managed/non display multicast market data, book snapshots, multicast recordings and packet retransmits, reference data, etc.

Lime Brokerage
Director of Engineering / Trade and Execution Services Division
November 2017 - February 2019

Primary Roles and Responsibilities:

Director for C++ engineering division - Market Data and Trading Server products as well as continuing to serve as lead architect for time frame listed below (July 2015-February 2019)

Lime Brokerage
Principal Software Engineer and Lead Architect
July 2015 - February 2019

Primary Roles and Responsibilities:

Lead architect, lead developer and mentor. Designed and implemented the company's next generation ultra-low latency market data product, "Citrius Direct". This project represented an entirely new code base designed specifically for this product (and subsequently migrated into existing products as needed). I worked almost exclusively on this project for about two years and was responsible for all architecture, design, documentation, and pretty much every line of code in the code base. The resulting work is an ultra low latency market data feed normalization product with an average market message latency of 750ns (from NIC to client API interface on fairly typical hardware). This projected included:

- **A complete networking library including ef_vi kernel bypass:**
  Authored a highly scalable networking library with support for kernel and ef_vi kernel bypass.

- **Custom memory allocators:**
  Developed a memory management library exploiting compile time information in combination with lock free programming and thread local storage to produce an incredibly efficient and scalable memory allocator.

- **Exchange market feed protocol parsers:**
  Developed a technique for parsing and marshaling the various financial exchanges' market feed protocols with ultra-low latency (5-10ns / message).

- **Super simple, compile time messaging transport:**
  Developed an incredibly easy to use messaging transport system for inter-process/inter-machine communication using compile time hashing to mark multi-part/multi-protocol messaging between ultra-low latency systems across a single transport stream.

- **Non locking data structures and novel custom shared locking techniques:**
  Developed a shared locking technique which exploits single thread per core architectures to heavily favor reader locks to eliminate contention for atomic share lock counts and authored a large library of lock free container classes.

- **Shared memory sockets for inter-process communications:**
  IPC sockets were way too slow for sub-micro second message processing across processes so I wrote a complete solution for inter-process communication which emulates UDP sockets using shared memory.

- **Ultra-low latency logging and software profiling library:**
  Employed my Glimpse library to profile and tune the entire project. Without the Glimpse tool there is no way to test/measure/tune/iterate with such precision. Using Glimpse I was able to make tiny changes in the code and quickly measure nanosecond level changes in performance over hundreds of millions of samples with ease allowing me to tune the product in a real world environment and with total confidence.

- **Extensive use of template meta-programming (when needed):**
  Made extensive use of TMP where needed to gain the efficiency needed to achieve sub-micro second/message performance. Such techniques were used in memory allocation/deallocation, message transports, protocol parsing, inter-machine endian ordering management, etc. In every such instance, great care was taken to ensure that the code complexities introduced by TMP were contained behind an easy to use, easy to maintain, facade.

Hydrolix
Principal Software Developer
November 2019 - October 2020

Primary Roles and Responsibilities:

Principal software developer for stealth mode start up.

- **M99b integer compression algorithm:**
  Invented a variation of my M99 compression algorithm, employing SIMD, to achieve high compression and high speed integer compression. Implemented many filters to improve compression based on apriori knowledge of the input data (sorted, strictly sorted, etc).

- **Metrics agent:**
  Developed process which collects metrics streamed from the main process. Forward metrics from there via HTTP requests to external consumer.

Saleae
Software Architect
February 2019 - August 2019

Primary Roles and Responsibilities:

Developing features for next generation of company's “Logic” application. Built module for on the fly storage and retrieval of recording of live streams of data (form hardware) for use in the “Logic” protocol analyzer software.

Viasat - (Formerly "Intelligent Compression Technologies")
Senior Software Engineer - Acceleration and Research Technologies Division
January 2002 - July 2015

Primary Roles and Responsibilities:

Senior Engineer responsible for algorithm development including all compression related software, various proprietary hashes, string pattern matching algorithms, language parsers, etc. I was generally responsible for any aspect of the division's products which have either high demands in time or space and for any features which require specific coding optimizations to meet such demands. I was also generally involved in most architectural issues. I was with this company (and the original start up) for well over a decade and was responsible for a lot of features and design decisions during that time.

- **Data Compression Suite:**
  Authored the company's compression library including custom implementations of LZ77, LZP, PPMd/PPMII, Huffman, massive scale delta compression, block delta and predictive block delta algorithms.

- **Delta Compression Algorithm and Patent:**
  Invented and authored the company's massive scale delta compression algorithm. A system of identifying similar (not necessarily duplicate) data from a massive repository with high speed and applying it as reference data to achieve dynamic delta compression. This compression engine serves as a core piece of the company's WAN accelerator. This algorithm was granted patent #US8010705B1 "Methods and systems for utilizing delta coding in acceleration proxy servers." The algorithm is the underlying delta compression engine used in Cisco's software solution for WAN acceleration.

- **"Associative" HTTP Prefetching:**
  Invented a neural networking approach for predictive HTTP object prefetching for use in HTTP acceleration over high latency networks. This work is capable of identifying previously recorded HTTP experiences from a massive database which might contain similarities to the current experience and then blending these data to produce predictions of future HTTP requests and responses.

- **Video Predictive Block Compression Algorithm:**
  Authored the company's 'predictive block delta' compression algorithm which is currently used for video over HTTP acceleration with high latency connections. This project involved designing a storage solution for use with flash RAM which evenly distributes writes across the physical medium to maximize the storage medium's useful lifespan, inventing algorithms to identify HTTP video streams in real time from any starting position, locating similar content in a local cache and maintaining HD video download rates using < 80KB of RAM to achieve near 100% compression rates over high latency networks.

- **HTTP Protocol Modeling and Compression:**
  Wrote the HTTP protocol parsing/modeling code which is at the core of the company's "Exede" HTTP web accelerator as well as in the Australian Government's NBN project for satellite internet.

- **Outlook Acceleration:**
  Studied and reverse engineered Microsoft's Outlook/MAPI protocol (long before they released the specification) and built the company's Outlook accelerator for WANs.

Portfolio: github, algorithms, research & inventions
Contributions to computer science, open source projects, independent research, compression and sorting algorithms

- **Glimpse - ultra low latency instrumentation/logging/data visualization:**
  [github private - available on request]

  Ultra high performance “type rich” application instrumentation and graphical analysis tools. This software can instrument C++ applications with incredibly low overhead (8ns-20ns per sample) and provide unbelievably rich, streaming, object oriented, real time instrumentation data which can be sampled, visualized, and mined by powerful visualization tools.

- **M99 - High performance BWT compressor:**
  github.com/michaelmaniscalco/m99

  I am the inventor of the M99 entropy encoding algorithm. Originally developed in 1999 as an entropy coder for the Burrows/Wheeler transform, this algorithm is a wavelet based entropy encoder specialized for encoding data which contains locally skewed symbol probabilities. It is a very simple, extremely fast, low memory encoding scheme which is highly effective on the right types of data (such as BWT data).

- **M03 - First and only “context aware” BWT compression algorithm:**
  github.com/michaelmaniscalco/m03

  I am the inventor of the M03 context based BWT compression algorithm. M03 is a progressive encoding scheme that achieves the highest compression of any generic Burrows/Wheeler based compressor. This is the only algorithm to date which can encode the Burrows Wheeler Transform with respect to the contexts contained in the original pre-transformed data. It is a fast, low memory compressor and has appeared in the paper "Post BWT Stages of the Burrows/Wheeler Compression Algorithm" by Dr Jeurgen Abel.

- **work contract library:**
  github.com/buildingcpp/system

  A simple, low latency, threading and asynchronous task management system. Benchmarks show that this unique approach to asynchronous tasks vastly outperforms existing methods and is especially well suited for use in ultra low latency systems.

- **network library:**
  github.com/buildingcpp/network

  A simple, easy to use network library designed to showcase “work contracts”.

- **MSufSort - suffix array construction algorithm:**
  github.com/michaelmaniscalco/msufsort

  MSufSort represented a large amount of my non-professional programming time. Over the years I have invented many specialized algorithms which have preserved MSufSort as the state of the art. When it was first introduced, MSufSort was 2-3x faster than the previous state of the art. The algorithm is described in the paper "An Efficient, Versatile Approach to Suffix Sorting", ACM Journal of Experimental Algorithmics Volume 12, Article 1.2 as well as in the paper “Faster Lightweight Suffix Array Construction” and is cited in numerous academic papers and journals . The most recent version of MSufSort (v4 alpha) achieves highly parallel suffix array construction which easily outperforms any existing suffix array construction solution by great margins. The work, however, remains incomplete and in alpha state.

- **RLE-EXP - Exponential run length encoding:**
  www.michael-maniscalco.com/download/10.1.1.12.5317.pdf

  Inventor of the RLE-EXP (exponential run length encoding algorithm). Since its first appearance in 2001 this simple enhancement on basic run length encoding has become a defacto standard encoding stage for many modern BWT compressors. RLE-EXP also appears in Dr Jeurgen Abel's paper "Improvements to the Burrows-Wheeler Compression Algorithm: After BWT Stages"

Publications:

- **An Efficient, Versatile Approach to Suffix Sorting:**
  Maniscalco & Puglisi - ACM Journal of Experimental Algorithmics Volume 12, Article 1.2
  http://www.michael-maniscalco.com/download/10.1.1.184.56.pdf

- **Faster Lightweight Suffix Array Construction:**
  Maniscalco & Puglisi - 17th Australasian Workshop on Combinatorial Algorithms (AWOCA'06)
  http://www.michael-maniscalco.com/download/10.1.1.183.9182.pdf

Patents:

- **Methods and systems for utilizing delta coding in acceleration proxy servers:**
  Patent #US8010705B1

  This patent covers the delta compression algorithms used in the Viasat WAN accelerator product. The basic algorithm is capable of identifying sources which are similar (not necessarily identical) with exceptionally high speed and accuracy. Similar sources are then used as reference dictionaries to achieve extremely high compression ratios.

- **Selective prefetch scanning:**
  Patent #US9407717B1

  This patent covers a method for scanning HTML and similar response data and predicting subsequent HTTP request produced by the browser which renders the data. These predictions are used to "pre-fetch" these HTTP requests and then position the response data closer to the requester in order to reduce page load times over high latency networks.