**PEDRO E. DOMINGUEZ**

Westport, Connecticut 06880203-252-9438pedroedr@gmail.com

**OBJECTIVE**

Hands-on C++ role developing low-latency high-throughput applications on UNIX-based platforms.

**EDUCATION**

BOSTON UNIVERSITY, College of Engineering - BOSTON, MA. Masters of Science, Electrical Engineering - 09/1998PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU - LIMA, PERU. Bachelors of Science, Physics – 03/1993**TECHNICAL SKILLS**

OPERATING SYSTEMS: Linux (RedHat, SUSE, CentOS), UNIX (Solaris 2.x, BSD, Darwin).LANGUAGES: C++11/14/17/20, C, Perl, Python, SQL, LabView, MATLAB.

PROGRAMMING & PROTOCOLS: Sockets, TCP/IP, UDP/IP, SunRPC, POSIX threads & IPC, Tibco, Talarian Smart Sockets, DHCP, Ethernet, ARP, XML, JSon, Qt, boost, RogueWave, Motif, Sybase, Oracle, ACTIV, OnixS FIX Engine.

HIGH-LEVEL: Distributed applications, trading systems, network equipment, test automation, ClearCase, SVN, p4, git.**WORK EXPERIENCE**

**Instinet (a Nomura Company), Latency Sensitive Electronic Trading (LSET)**Vice President (07/2020-11/2023)

* Developed components and functionality in C++20 on a Linux/CentOS platform, to support Edge, Instinet’s next generation market access suite. Focused on implementing gateways to upstream legacy systems, and to exchanges. Leverage on the OnixS FIX Engine product for FIX-based flows, both internal and external. Ongoing project to develop a normalized gateway framework handling exchange-specific native protocols, based on Celoxica’s order entry GXA product.  
  Development environment implements industry’s best practices for testability, code structure, and workflow (with CLion IDE, git, TeamCity CICD, gtest, cmake).
* Developed C++17/Linux functionality to support NXTBook, a market data broker/engine subscribed to feeds from multiple US and EU equities exchanges, that publishes L2 books, trades, imbalances. Used by groups within Instinet as well as external clients. The underlying infrastructure is based on Celoxica’s GMAC 2.1 product, which consists of an FPGA accelerated NIC, a runtime library creating multiple isolated pinned feed handler threads and a shared memory queue, and an API to consume normalized data from it; L2 snapshot processing functionality also supported. Average latency from tick to order book update is 1.5 microseconds.
* Developed C++20/Linux functionality to support UDC, a network packet capture and replay engine. The underlying infrastructure relies on Napatech’s SmartNIC FPGA, its Link-Capture software and NTAPI.

**Morgan Stanley, Multi-asset Algorithmic Trading**Vice President (09/2014-03/2020)

* Developed C++14/Linux infrastructure and app-level functionality to support business requirements for all asset-specific instances of VSORT, Morgan Stanley’s low latency smart order router. Emphasis is on the **cash equities** MSET asset class instances which depending on market activity, fill in excess of 1 billion shares daily.

A given VSORT instance is configured to receive client orders for a sub-set of symbols, and to process them in multiple isolated pinned order-handling threads. Interaction with peripheral processes is via multicast.

* Developed various generic framework components to improve VSORT’s fill rate, latency, reliability/availability:  
  (1) Aggressive Quote Dropping, which keeps track of attempted L2 and top-of-the-book/Montage quotes; the objective being to direct new orders to price levels deeper into the L2 book, by temporarily excluding already attempted better-priced quotes. Implementation leverages on thread local data and custom STL allocators.  
  (2) SymbolProfile, which maintains per symbol/product general data, and also specific to each configured exchange. A profile instance is directly accessible (i.e. no lookups) in the critical path and can be modified at runtime by production support teams as needed. Allocation of redundant/duplicate profiles is avoided by associating a given one with all symbols sharing the same data. Copy-on-write storage in thread local maps.  
  (3) Asynchronous Worker, which off-loads expensive operations from VSORT’s order-handling threads’ critical path, to an un-pinned worker thread. E.g. an order received for a new product that triggers a db lookup and which will require a new market data subscription. Requests/responses exchanged via lock-free SPSC queues.  
  (4) SyncTake, which manages the dispatch time of the multiple exchange-specific slices resulting from models/algorithms being applied to a client order. The objective is for each of these slices to reach its corresponding market, all at the same time, thus avoiding self-arbitrage; the challenge being VSORT is not co-located with all trade-able exchanges. Implementation is based on a single dispatch thread holding slices from multiple order-handling threads, and passed to it via a lock-free MPSC queue.

(5) Refactored VSORT’s real-time market data framework, which leverages on in-house developed infrastructure libraries that abstract an interface to a dedicated hardware-based ticker plant. This enabled a VSORT instance to run multiple market data sessions/threads, by implementing a 1-to-N association of each configured one, to a configurable group of order-handling threads. Also, the consumption of snapped book updates was optimized by storing them in a thread local over-writable per-evaluation place-holder book.

* Implemented functionality for a multi-asset FPGA-based Market-Making framework (aka Proxima). Current prod set up subscribes to Xetra for ETF market data, and to Eurex for futures, making market for ETFs on the former.

**WELLS FARGO Securities, Equities Trading**Applications Systems Engineer (05/2012-09/2014)

* Developed infrastructure for NOVA, a low-latency C++11/Linux Order Management system based on memory-mapped-file state replication; also provided daily support to production desks and UAT teams.
* Re-designed and implemented a QuickFix-based order entry framework to support handling transaction engines from multiple distributed clusters, with emphasis on concurrency and recovery.
* Worked on optimizing NOVA’s market data server, which consumes data from ACTIV’s Content Gateway ticker plant; updates are processed from multiple threads using lock free queues, and data persisted in shared memory for consumption by local downstream applications.

**LAVA Trading, CITIGROUP** **Corporate Investment Bank**Vice President, Technical Specialist (08/2011-05/2012)

* Worked on the re-design of LAVA’s FLOW ECN; migrated Windows-dependent C++ code to Linux (RHEL6).
* Enhanced functionality of FLOW’s Trade Reporting Application by enabling multiple connections to NASDAQ ACT and adding fault tolerant functionality.

**UBS Investment Bank, Electronic Volatility Trading, Equity Options Market Making**Associate Director (06/2008-07/2011)

* Designed, implemented and deployed EVT's valuation/pricing/risk engine/server, TVGEN. It is a multi-threaded C++ application with currently 60+ instances of it running in various Linux machines, each instance handling a configurable set of underlyings/options. TVGEN subscribes to options and underlying market data feeds, and for each calculates a consolidated price and an implied volatility; these are used to fit the term's volatility smile curve from which theoretical (quoting) option prices are calculated. It’s threading/queuing model allows for:

(1) optimal throughput, by performing parallel calculations/fits for options in different stages of processing or under different terms or underlyings.

(2) handling of user-generated asynchronous term/underlying-level msgs without affecting ongoing calculations.

(3) efficient calculation of NBBO using direct market data from various exchanges.

* Developed a multi-threaded C++/Linux quote/order/auction entry/management application to make market in NASDAQ PHLX options exchange; acts as a gateway for instances of EVT’s central quoting engine (OGEN) by:

(1) implementing NASDAQ's SoupBinTCP and SQF version 6 protocols to reliably handle messages, e.g. block quote entry, underlying purges, acknowledgements, notifications, auction responses.

(2) managing multiple connections per matching engine by prioritizing transactions and ensuring synchronicity.

(3) processing clearing trade updates (fills, corrections, cancels/busts) using NASDAQ´s CTI protocol.

(4) receiving quote/order/auction related commands from OGEN clients and sending acknowledgements to it based on the response from the exchange, nature of the transaction and current state of instruments involved.

* Worked on a generic C++/Linux feed handler framework to receive binary market data using venue specific protocols, decode it and publish it to upstream engines. Implemented adapters for BOX (Boston Options Exchange) HSVF, and for NASDAQ ITCH 4.1.
* Developed generic infinite depth order book to handle insertion, updates, execution deltas, removals, uncrossing; used by multicast market data feed handlers like ITCH, ICE and DirectEdge.

**BEAR STEARNS, Equity Derivatives Technology, Index Arbitrage**Vice President (04/2007-06/2008)

* Develop new applications, enhance existing functionality and provide daily production support to the front office RAPTOR (Real-Time Position Tracking & Order Routing)distributed C++real-time basket trading system, which is used by the index arbitrage and volatility desks. Some infrastructural enhancements worked on were:

(1) Functionality to its Talarian-based middleware library to allow parallel processing of quote-type messages.

(2) Observer pattern with queuing and multithreaded event processing to distribute load from mkt data callbacks

(3) Integration of Trolltech’s Qt SQL module into RAPTOR’s database access Linux-based libraries.

* Designed, developed and released into production a stand-alone C++ Solaris-based server process -the Price Cache Adapter (PCA)- as part of RAPTOR’s next generation infrastructure. The PCA creates a load-balanced number of Tibco EMS session threads to receive real-time binary quotes, both market (Reuters Triarch RFA, Wombat MAMDA) and calculated (proprietary predictive algorithms), published by several remote linux-based PriceEngine processes. Upstream RAPTOR processes then consume these quotes from shared memory.

**CITIGROUP**, **Corporate Investment Bank, Equity Desk Trading System (EDTS)**Assistant Vice President, Technical Specialist (08/2003-04/2007)

* Lead engineer on the re-architecture and maintenance of Citigroup’s proprietary UNIX networking real-time Messaging product: Common Communication System (CCS). Event-based point-to-point persistent C++ distributed framework integrated into trading systems; based on SunRPC over TCP/IP.

**LUCENT TECHNOLOGIES INC**, **Network Systems Group.**Member of Technical Staff, Network Systems Engineer (08/1998-04/2003)

* Developed test software (C++, UNIX Solaris 2.x), automated test equipment instrumentation software (drivers, VXI, RS232, GPIB IEEE 488.2 compliant) and networking applications (sockets API, Sun RPC, TCP/IP, UDP/IP, Ethernet, ARP, ICMP, DHCP Server) to enable connectivity between test servers and units under test (running VxWorks-based firmware) and perform measurements.