Scaling Financial Transaction using 0MQ and JSON

Moving from 03 -> 11 -> 14 and other learnings.

Background and History

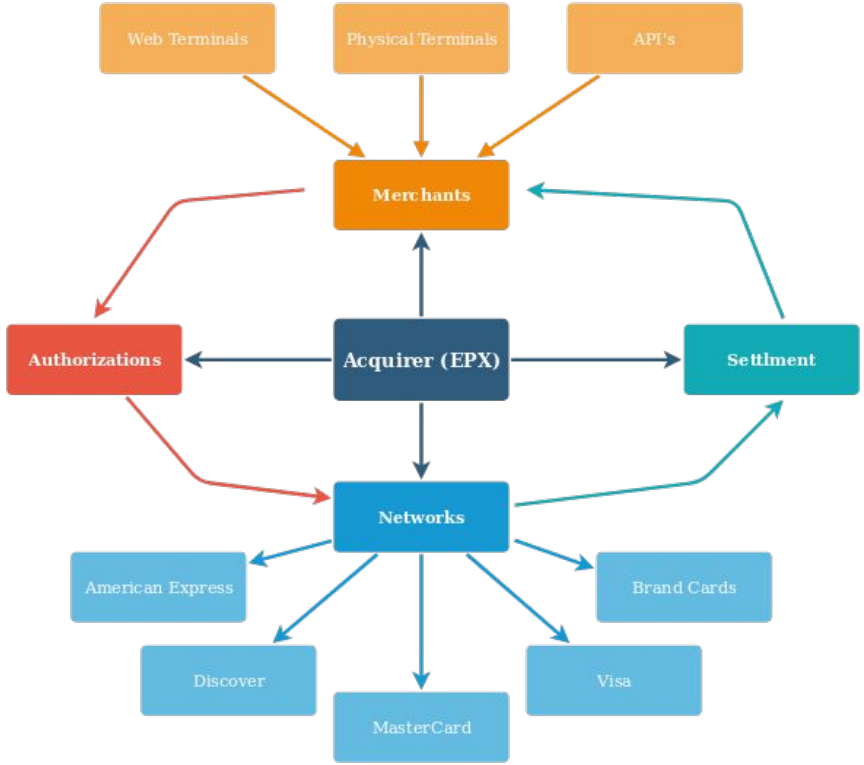
- Started developing in C++ in 1998 with Borland C++ Builder
- Financial Simulations in C++/MFC until 2016.
- Been working with C++ and in Linux since then for EPX.
- North American Bancard acquired EPX in 2014
- We process over $50 billion dollars a year in transactions.

Housekeeping and Disclosures

- Definitely a lessons learned talk.
- Map of transactions and how they flow.
- As a development group we choose smaller (header only, or similar) stable open source libraries.
- 0MQ is primarily C, however we use the the CPPZMQ interface.
- I/We are not 0MQ(ZMQ) Experts
- Our backend libraries are C++.
- There are open standards on Credit Cards such as the EMV standard. http://emvco.com
- No PCI Compliance issues here.

Building for Scalability Breadth, Speed, Stability

- Moving parts of the Processing System.
- Many have their own protocol. EBCDIC anyone?
- Some are old, some are new, some are BETA.
- Card standards have more specifications then code.



The Legacy -> Moving Forward

- PowerBuilder
- Windows Based A/B Redundancy
- Normalization anyone?
- Almost never good to rewrite, but sometimes that's what has to be done for longevity of a product or line of business.
- Technical debt is due to outdated language tools and legacy implementations.
- Must prevent new bugs as much as possible however its a rewrite in small testable checks.
- Testing is key to finding newly implemented bugs!

The Tech: 0MQ & JSON
Instead of Sockets and XML

- Why 0MQ
  - Stability
  - Brokerless
  - C0MQ / CPPZMQ
- Why JSON (instead of XML)
  - Readability
  - Portability
  - Size
- Other Synergies
  - 0MQ uses char* by default.
  - Oracle OCCI and data buffers; everything can be a string.

JSON for Modern C++

What if JSON was part of modern C++?

This is an animated GIF. Please wait for a feature slideshow.

CentOS Default Compiler, Oracle and ABI

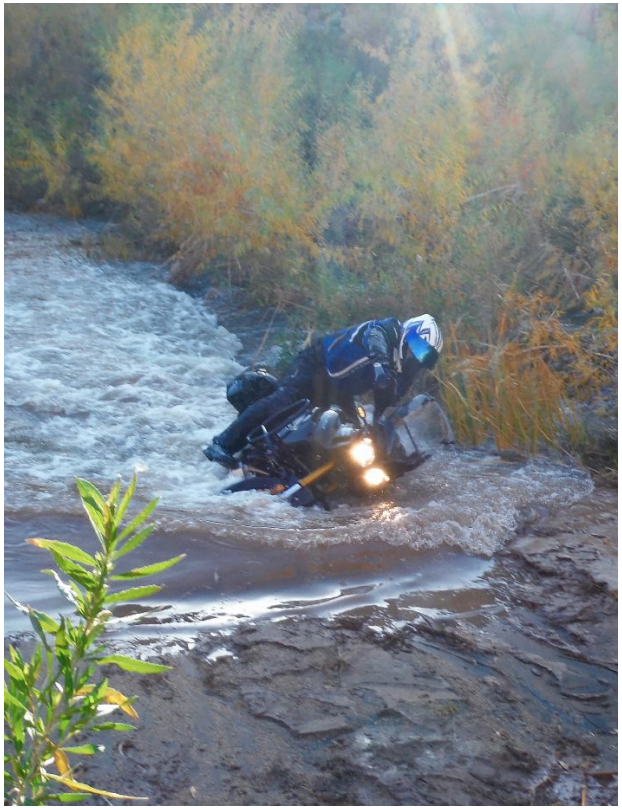
To move forward we needed a C++ 11 compiler.

CentOS6 by default didn't have one.

In our makefile you will find:

CXX = g++ -D_GLIBCXX_USE_CXX11_ABI=0

Oracle upgrade caused ABI issues.



How much smaller is the JSON?

```xml
<request>
<params>
<param name="guid">09EFMDSREG46EJKEREX62TGF1F7</param>
<param name="cardType">A</param>
<param name="processingDate">10/23/2018</param>
<param name="transactionType">15</param>
<param name="authorizationAmount">995.00</param>
<param name="inventoryCode">0L</param>
<param name="verifyMerchantCode">7991</param>
<param name="wasSwiped">Y</param>
<param name="detailResult">N</param>
</params>
</request>
```

```json
{
    "guid": "09EFMDSREG46EJKEREX62TGF1F7",
    "cardType": "A",
    "processingDate": "10/23/2018",
    "transactionType": 15,
    "authorizationAmount": 995.00,
    "inventoryCode": "0L",
    "verifyMerchantCode": 7991,
    "wasSwiped": "Y",
    "detailResult": "N"
}
```

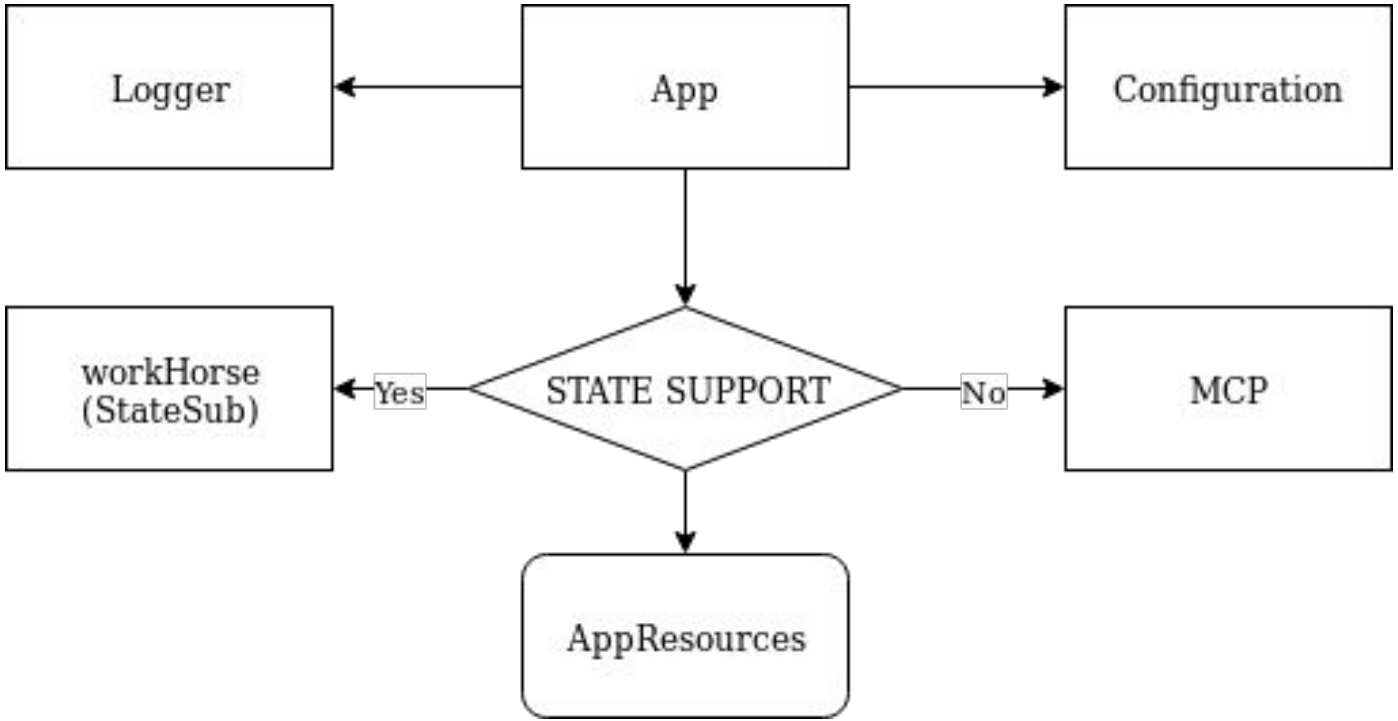
413 vs 262 = 35% reduction

Benefits of JSON for Modern C++

- Requires only C++ 11
- Header only.
- Easy to read and use.
- JSON Portability; BSON in BETA
- Better type management.
- https://github.com/nlohmann/json

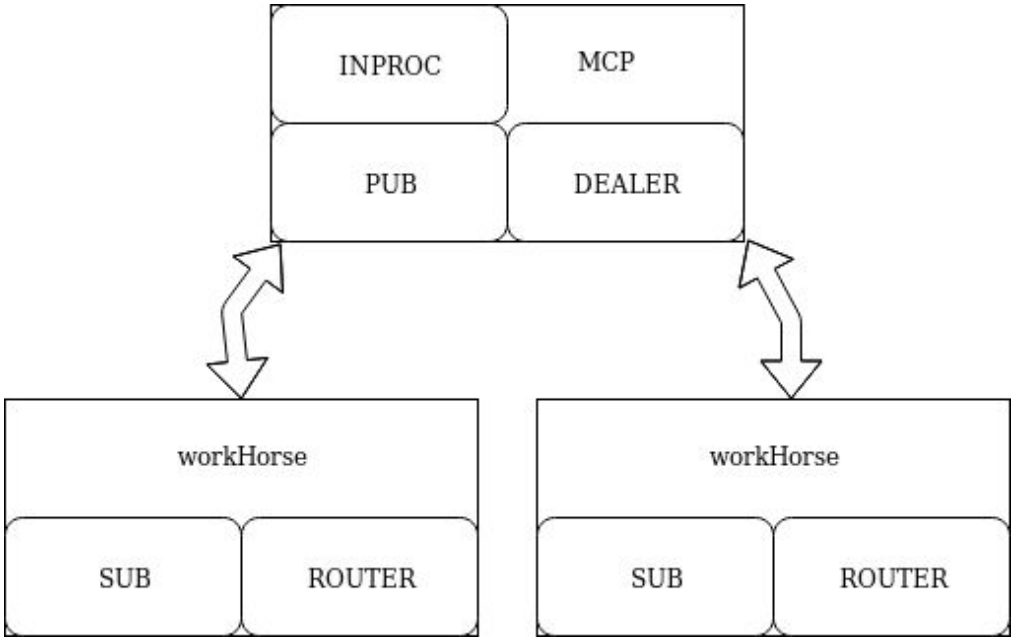
```
{
    "guid": "09EFMDSREG46EJKEREX62TGF1F7",
    "cardType": "A",
    "processingDate": "10/23/2018",
    "transactionType": 15,
    "authorizationAmount": 995.00,
    "inventoryCode": "0L",
    "verifyMerchantCode": 7991,
    "wasSwiped": "Y",
    "detailResult": "N"
}
```

Application and Class Layout



0MQ and the Patterns

- DEALER/ROUTER
- PUB/SUB
- INPROC



0MQ - ROUTER of DEALER/ROUTER

```
13  //--------------------------------------------------------------------------
14  ZmqWorkHorse::ZmqWorkHorse( const json& config, zmq::context_t& context ) :
15      context( context ), socket( context, ZMQ_ROUTER ) {
16      string connection{config[ "server" ][ "ip" ].get< string >( ) + ":" +
17          to_string( config[ "server" ][ "zmqPort" ].get< ushort >( ) )};
18      socket.bind( "tcp://" + connection );
19      logger = Logger::getLogger( );
20
21      logger->save( logInfo( __func__ ), "Listening on " + connection, IS_MAIN );
22  }
```

```
172  //-----------------------------------------------------------------------
173  void Cortex::sendAction( const json& js ) const {
174      // Build action
175      AppIdentity id = js;
176      {
177          shared_lock< st_mtx > lck( appMapMtx );
178          if ( appMap.find( id ) == appMap.end( ) ) {
179              logger->warn( logInfo( __func__ ), "Invalid AppIdentity: " +
180                  id.getAppId( ), IS_MAIN );
181              return; // Not a valid app
182          }
183      }
184
185      zmq::socket_t socket( context, ZMQ_DEALER );
186      socket.connect( "tcp://" + id.getConnectionInfo( ) );
187
188      string temp{js.dump( )};
189      zmq::message_t message( temp.data( ), temp.length( ) );
190      socket.send( message );
```

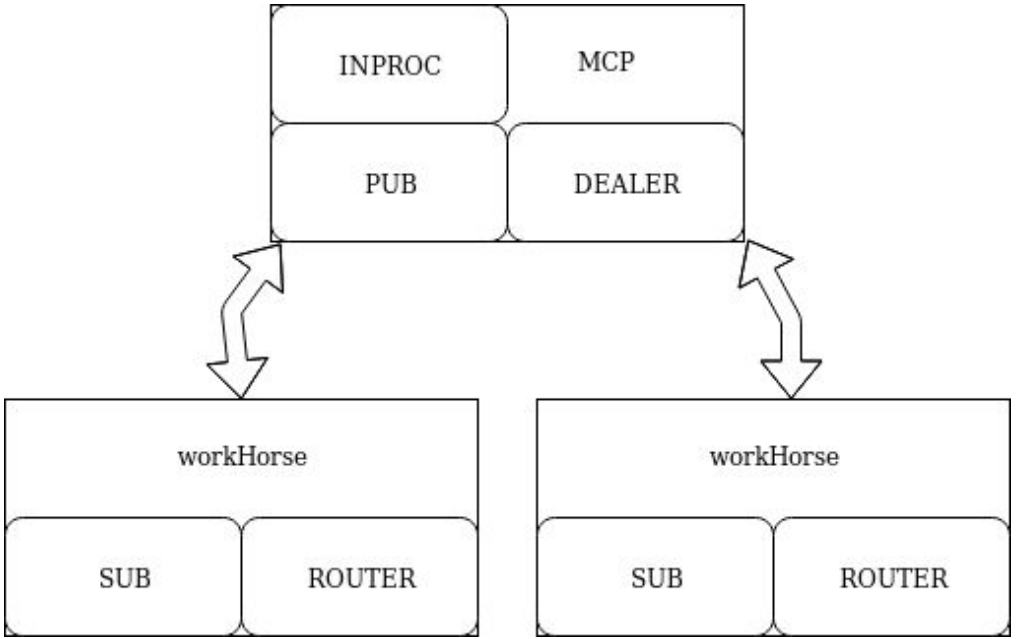
You have become better at C++ (14)

To and From JSON

```
243  //-----------------------------------------------------------------------
244  void to_json( json& j, const AppIdentity& id ) {
245      j[ "appName" ] = toString( id.name );
246      j[ "ip" ] = id.ip;
247      j[ "port" ] = id.port;
248  }
249
250  //-----------------------------------------------------------------------
251  void from_json( const json& j, AppIdentity& id ) {
252      id.name = j.at( "appName" ).get< AppType >( );
253      id.ip = j.at( "ip" ).get< string >( );
254      id.port = j.at( "port" ).get< ushort >( );
255  }
```

0MQ and the Patterns

- ~~DEALER/ROUTER~~
- PUB/SUB
- INPROC



0MQ - SUB of PUB/SUB

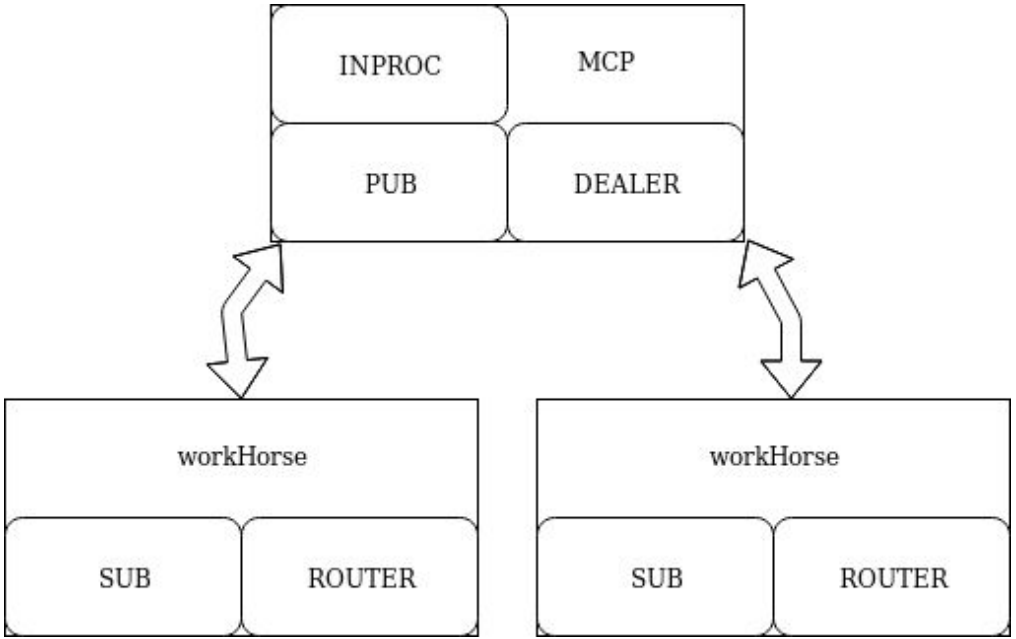
```
16  //---------------------------------------------------------------------------
17  void StateSub::subscribe( const json& config, ushort level ) {
18      zmq::socket_t socket( context, ZMQ_SUB );
19      if ( level > StateLevel::MAX_LEVEL ) {
20          level = StateLevel::MAX_LEVEL;
21      }
22      string sLevel{to_string( level )};
23      socket.setsockopt( ZMQ_SUBSCRIBE, sLevel.data( ), sLevel.size( ) );
24      socket.connect( "tcp://" + config.at( "control" ).at( "host" ).get< string >( ) + ":" +
25          to_string( config.at( "control" ).at( "subPort" ).get< ushort >( ) ) );
26
27      while ( true ) {
28          try {
29              // Get broadcast message
30              zmq::message_t address, message;
31              socket.recv( &address );
32              socket.recv( &message );
33              string sMessage( static_cast< char* >( message.data( ) ), message.size( ) );
```

0MQ - PUB of PUB/SUB

```
97  // Cortex::publisher
98              switch ( appId.name ) {
99                  case AppType::mcp: ···
102                 case AppType::workHorse:
103                     level = StateLevel::RESOURCE;
104                     break;
105                 default: ···
107             }
108             for ( ushort i{level}; i ≤ MAX_LEVEL; ++i ) {
109                 string tmp{to_string( i )};
110                 zmq::message_t addr( tmp.data( ), tmp.length( ) );
111                 zmq::message_t msg;
112                 msg.copy( &inMsg );
113
114                 publisher.send( addr, ZMQ_SNDMORE );
115                 publisher.send( msg );
116             }
```

0MQ and the Patterns

- ~~DEALER/ROUTER~~
- ~~PUB/SUB~~
- INPROC



INPROC Example

```
81  //-------------------------------------------------------------------------
82  void Cortex::publisher( ) {
83      zmq::socket_t inSock( context, ZMQ_PULL );
84      zmq::socket_t publisher( context, ZMQ_PUB );
85      inSock.bind( "inproc://publisher" );
86      publisher.bind( "tcp://" + config.at( "server" ).at( "ip" ).get< string >( ) + ":" +
87          to_string( config.at( "server" ).at( "pubPort" ).get< ushort >( ) ) );
88
89      while ( true ) {
90          try {
91              zmq::message_t inMsg;
92              inSock.recv( &inMsg );
```

```
124  //-----------------------------------------------------------------------
125  void Cortex::broadcastAll( ) const {
126      zmq::socket_t inSock( context, ZMQ_PUSH );
127      inSock.connect( "inproc://publisher" );
128      shared_lock< st_mtx > lck( appMapMtx );
129      for ( const auto& app : appMap ) {
130          json id = app.first;
131          json res = app.second;
132          string str{merge( id, res ).dump( )};
133
134          zmq::message_t message( str.data( ), str.length( ) );
135          inSock.send( message );
136      }
137  }
```

0MQ and the Patterns

- ~~DEALER/ROUTER~~
- ~~PUB/SUB~~
- ~~INPROC~~

DEMO

Performance Improvements

DB Caching Application - 90 minute update down to 10 minutes.

Authorization Transfer - 5x speed-up

Encrypt/Decrypt Routines - 3x speed-up

Platform Redundancy and Scalability

Code, Slide and Thanks

https://github.com/kevinbcarpenter/jz18sub

Kent Glenn, Jeff Jacob, Mithu Gansen, Jermey Daley