Useful C++ Software libraries

Modification history

- 18-12-2014 Guus Bonnema: adding problem definition, history and conclusion (summary).
- 03-12-2014 Stefan Versluys: document creation and subsequent modification

Problem definition

We need to know what libraries to use next to the user interface ~~FLTK~~ `Qt`. This is a comparison of tools that can be used for the project.

The product is a standalone application and data is currently `JSON` structured. The main properties that this application will need are multi threading or processing, inter process communication, serialization, regular expressions, streaming, exception handling.

Conclusion

The package `Boost` is the recommended set of libraries to use for features our user interface does not provide. See final conclusion at the end of this readme.

Comparison of libraries that could be used.

- POCO C++
- NCBI
- Boost
- Protocol Buffers
- MessagePack
- Avro
- ZeroMQ

POCO C++

The POCO C++ libraries focus on internet-centric applications. Released under the Boost Software License. >Conclusion: Not very useful because its focus differs from the needs for this project.

NCBI

Library supports multi threading, serialization, ipc (sockets & pipes), regex. >Conclusion: Useful but need to add additional libraries to support JSON. >Documentation is poor and only a few who referenced to this toolkit, seems >to be non popular.

Boost

Boost is a generic library with many features. It has libraries for serialization, multi threading. Boost also includes a basic JSON parser, IPC and much more.

> Conclusion: Boost is a popular set of high quality libraries. If it comes to multi threading or processing Boost is a good choice. Many simple tryout examples and good documentation.

Protocol Buffers

Google"s Protocol Buffers are used for serializing structured data based on RPC. The structure or message is described by a interface design language (IDL). Its main goal is communication over wire and store data in a binary form.

> Conclusion:Protocol Buffers is simple and efficient but purely used for serializing. The idl *proto files* need to be compiled with a specific compiler called *protoc*.

Avro

Apache Avro is pure serialization based on RPC. Avro is more like a hybrid it has the same flexibility as Protocol Buffers but the schema is inside the header and doesn't need a compiler. Schema's are in JSON format.

> Conclusion: Ideal when there's a need for strongly typed formats combined with flexibility. Schema's can be made in pure JSON but gives some overhead because it is always part of a message it represents.

MessagePack

MessagePack is like JSON but smaller and faster

> Conclusion: No type checking schemas but a perfect drop-in replacement if the API is already using JSON for streaming and storage.

ZeroMQ

This is a networking library, so purely used for communication. In-process, interprocess, TCP and multicast.

Conclusion: A pure communication library but easy to use.

Final Conclusion

There is no *best library* it all depends on the needs. Boost covers a wide range of good solutions Because this project is standalone and the main needs are serialization for local storage, communication between processes on the same machine which are related to each other. If communication between processes is local and based on C++ objects then Protocol Buffers doesn't add advantages. Also multi threading, array's, regular expressions are some of the mechanisms that this project will possible need. So Boost is the most appropriate choice because it covers many of those mechanisms. It is always possible to use Avro or MessagePack next to Boost because these are focused on serialization based on JSON.

References:

- C++ Libraries
- Serialization tools
- Pipes