Low Latency Abax C++ Container Class Library

Exeray Inc

Contents

Low Latency Computing

Latency is a time delay between the cause and the effect of some physical change in the system being observed. Ensuring low latency in systems has significant business values.

Gaming and Animation

Computer online games are very sensitive to latency since fast response times to new events occurring during a game session are rewarded while slow response times may carry penalties. High availability and low-latency storage are critical to effective data management in gaming, whether it is supporting in-game player and session data, or profiles and the player community forums. Irrespective of the game' s complexity, the management of the architecture needs to be simple and support the rapid growth that can be needed as a game gains popularity, something we have seen happen many times through social media.

Financial Markets

Minimizing latency is currently of interest in the capital markets particularly where algorithmic trading is used to process market updates and turn around orders within milliseconds. Low latency trading refers to the network connections used by financial institutions to connect to stock exchanges and Electronic communication networks to execute financial transactions. Latency is measured based on three components: the time it takes for 1) information to reach the trader, 2) the trading algorithms to analyze the information, and 3) the generated action to reach the exchange and get implemented. With the spread of computerized trading, electronic trading now makes up 60% to 70% of the daily volume on

the NYSE and algorithmic trading close to 35%. Trading using computers has developed to the point where millisecond improvements in network speeds offer a competitive advantage for financial institutions.

Exeray Abax Low Latency C++ Classes

Storage and retrieval of data is one of the fundamental aspects of any computer program. Containers are a type of are a type of data structure that allow the storage of a collection of data and retrieval of the data. Design and implementation of the containers directly impacts the efficiency of the containers.

Traditional container classes such as Standard Template Library have been implemented with tree structures to store data elements. The asymptotic complexity of the operations that can be applied to associative containers, where key-value pairs are stored, are as follows:

Operation	Complexity
Searching for an element	O(log n)
Inserting a new element	O(log n)
Incrementing/decrementing an iterator	O(log n)
Removing a single element	O(log n)

Because every millisecond is important to low latency computing, we have redesigned and implemented container classes with flat array, instead of tree data structure. Flat arrays have excellent property in data locality and caching

performance. Combined with efficient algorithms, our associative container classes have the following drastically improved time complexity:

Operation	Complexity
Searching for an element	O(1)
Inserting a new element	O(log n)
Incrementing/decrementing an iterator	O(1)
Removing a single element	O(1)

Our associative containers provide constant search time while maintaining ordering of data elements in the containers. Iterating over ordered elements and removing elements are performed with extremely low latency.

The following is a list Abax C++ Container Classes:

Container	Description
AbaxMap	Hashed and sorted key-value store
AbaxMultiMap	Hashed and sorted multiple key-value store
AbaxSet	Hashed and sorted key store
AbaxMultiSet	Hashed and sorted multi-key store
AbaxHashMap	Hash table of key-value store
AbaxHashSet	Hash table of key store
AbaxHashList	Linked list with hash access, insertion and key order
AbaxCounter	Key store with counts of the keys, key and count order
AbaxPriorityQueue	Priority queue
AbaxQueue	FIFO queue class
AbaxStack	LIFO stack class

AbaxVector	Vector class
AbaxList	Single linked list
AbaxGraph	Dynamic undirected and directed graph
AbaxReadWriteMutex	Read and write mutex for concurrent threads

Use Cases of Abax Containers

Use Case One

C++ developers can just use AbaxMap to replace std::map class in programming:

```
// map<string, string>  oldmap;

// oldmap.insert( pair<string, string>("AAPL",
"close:128.20;low:125.29,high:129.42");

// oldmap.insert( pair<string, string>("YHOO",
"close:43.65;low:45.19,high:41.45");

AbaxMap<AbaxString, AbaxString> map;

map.addKeyValue("AAPL",
"close:128.20;low:125.29,high:129.42");

map.addKeyValue("YHOO",
"close:43.65;low:45.19,high:41.45");


Then retrieving value for a key is with O(1) low latency:

AbaxString value;

bool rc = getValue( "YHOO", value );
```

Use Case Two

Very often applications are required to monitor the data elements that receive the most interests, such as quotes of stock price in a trading environment. In this case, AbaxCounter is the best container to use:

```
AbaxCounter<AbaxString> counter;
counter.addKey("IBM");
counter.addKey("ORCL");
counter.addKey("IBM");
counter.addKey("YHOO");
counter.addKey("ORCL");
counter.addKey("IBM");
AbaxCounterCountReverseIterator<AbaxString>
iter( &counter );
while ( iter.hasNext() ) {
        const AbaxString &s = iter.next( );
       abaxcout << s << abaxendl;
}
```

The output is as follows where counts of each element is printed with descending order:

```
3,IBM
2,ORCL
1,YHOO
```

How To Download Abax Enabler Class Library

You can visit http://www.exeray.com/product.php or http://github.com/exeray/abaxenabler to download the Linux 64 binary version of the Abax class library. These two web sites also have the user manual for developers to download. Example program is also provided in the downloadable package.