Wrocław University of Technology
Faculty of Electronics

Milestone 3
Component COMPUTATION

C++ multithreading overview

Using Visual Studio 2012 and Boost library

Author:
Rafał Maćkowski
2014-11-13

Reviewer:
Bogumił Chojnowski

Status:
Approved

Introduction

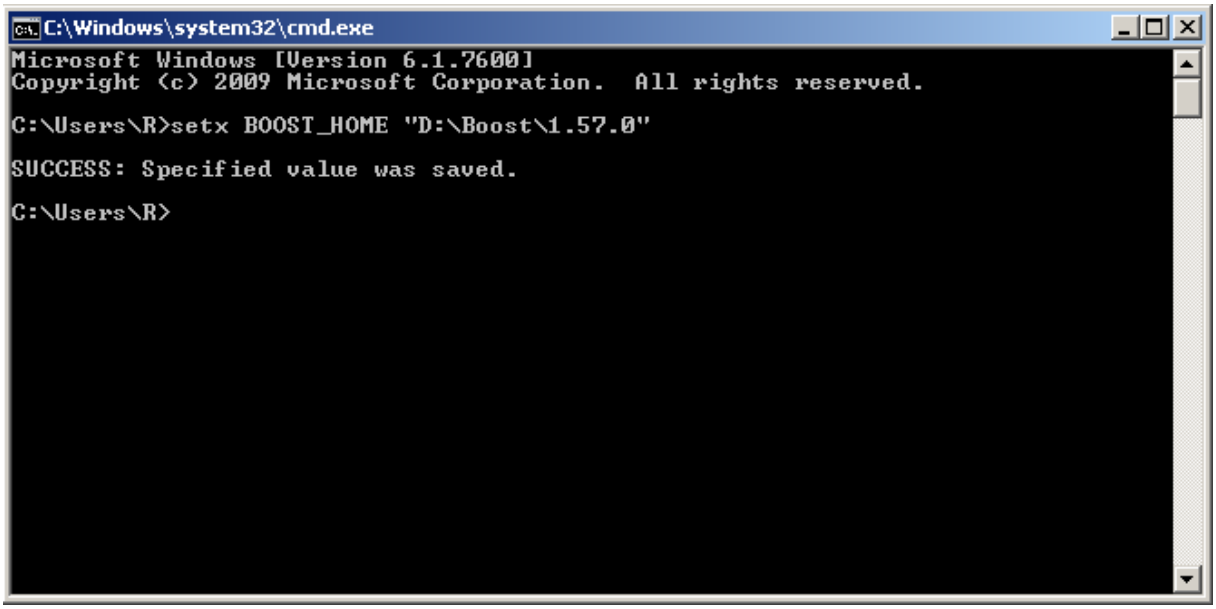
In this document I will try to explain how to install and use Boost library in Visual Studio 2012. Everything described below was tested using PC with installed Windows 7 and Visual Studio 2012 Express Edition.

Why Boost library?

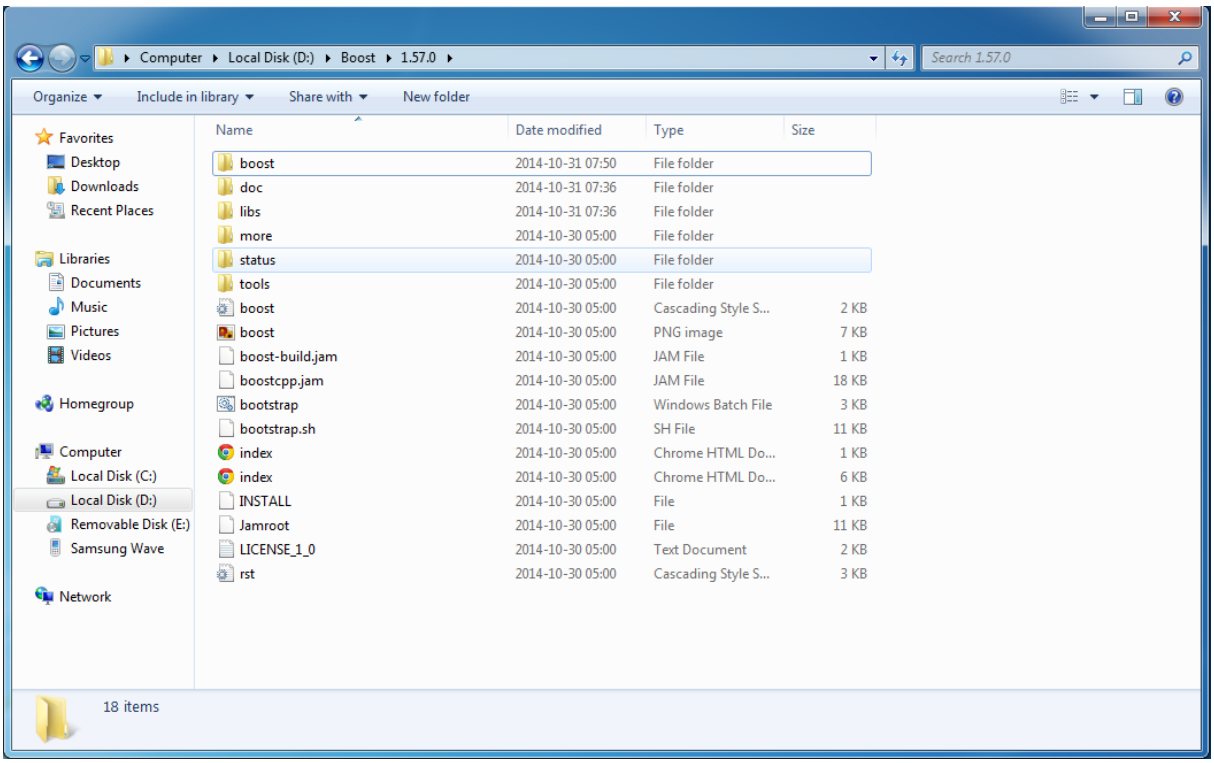
If we want to improve realism, as much as it is possible and keep computations in real time, we should use multithreading. There are many libraries for managing threads. The best suited candidate is Boost library. The library consist a lot of mechanisms for synchronization and managing the threads, so it’s very complete. Boost is developed for long time. Thanks to this we can assume, that it is well tested. Also good documentation is big advantage. Thanks to popularity of this library, it is very easy to find examples of code and solution for common problems.

Building Boost library

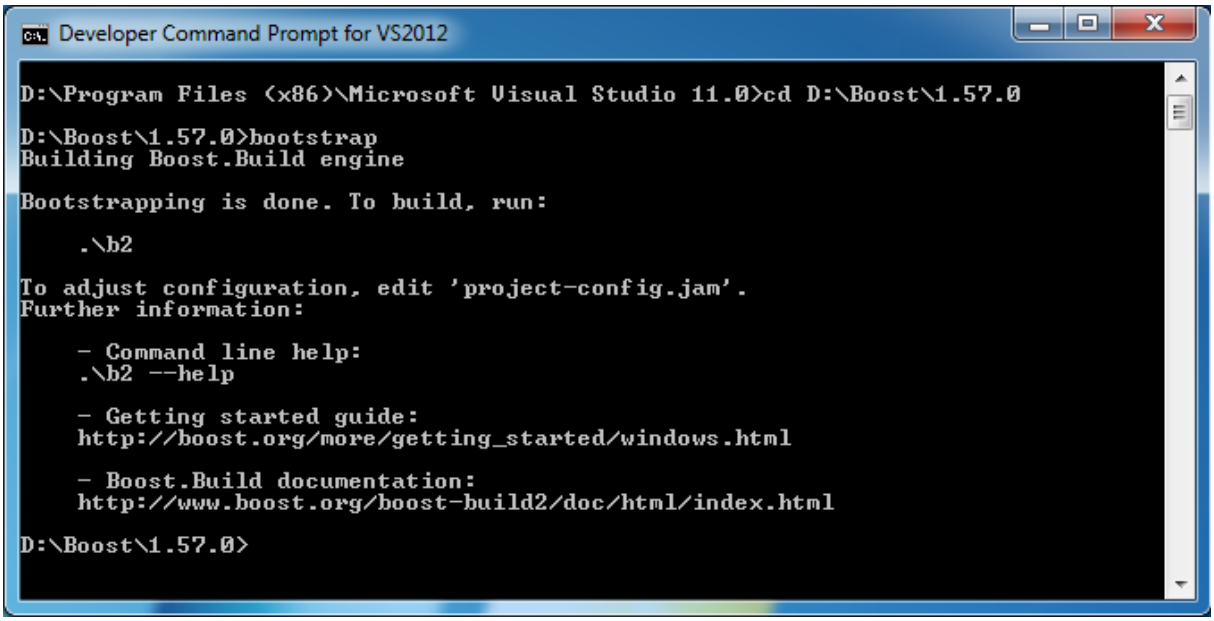
- Create folder to keep Boost files (in my example it was D:\Boost\1.57.0)
- Set path variable for just created folder “setx BOOST_HOME“D:\Boost\1.57.0”



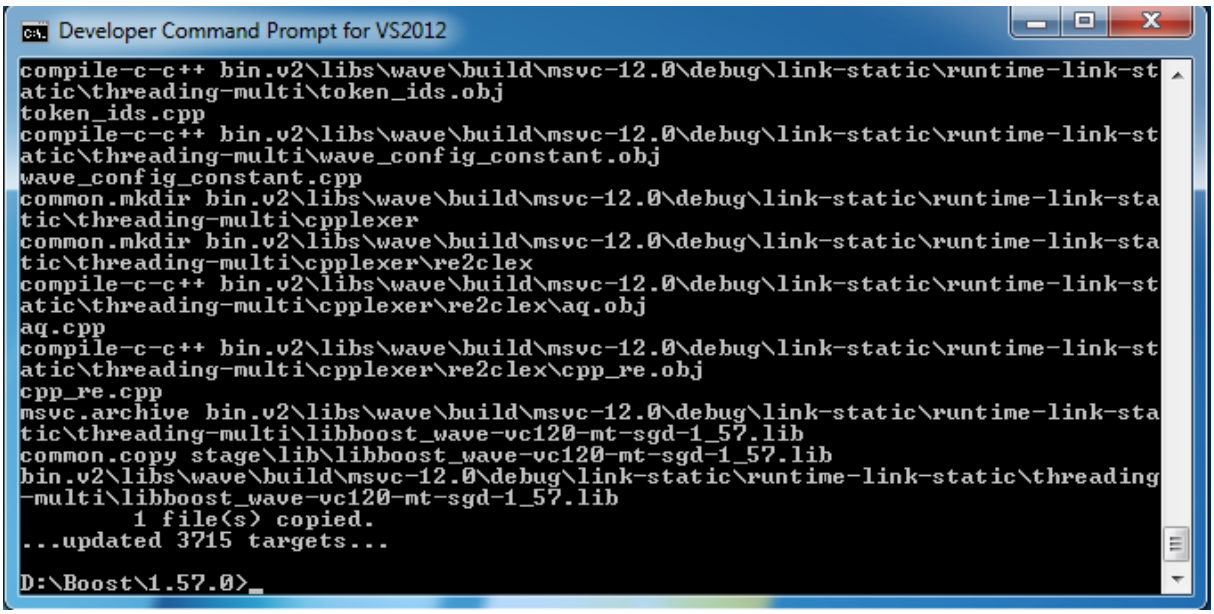
- Download Boost library (1.57.0) from http://sourceforge.net/projects/boost/files/boost/1.57.0/boost_1_57_0.zip/download
- Extract downloaded file and copy content of boost_1_57_0 directory to your BOOST_HOME folder



- The same way as you run “cmd”, run “Developer Command Prompt for VS2012”
- Using “cd” command move to your BOOST_HOME and execute command “bootstrap”

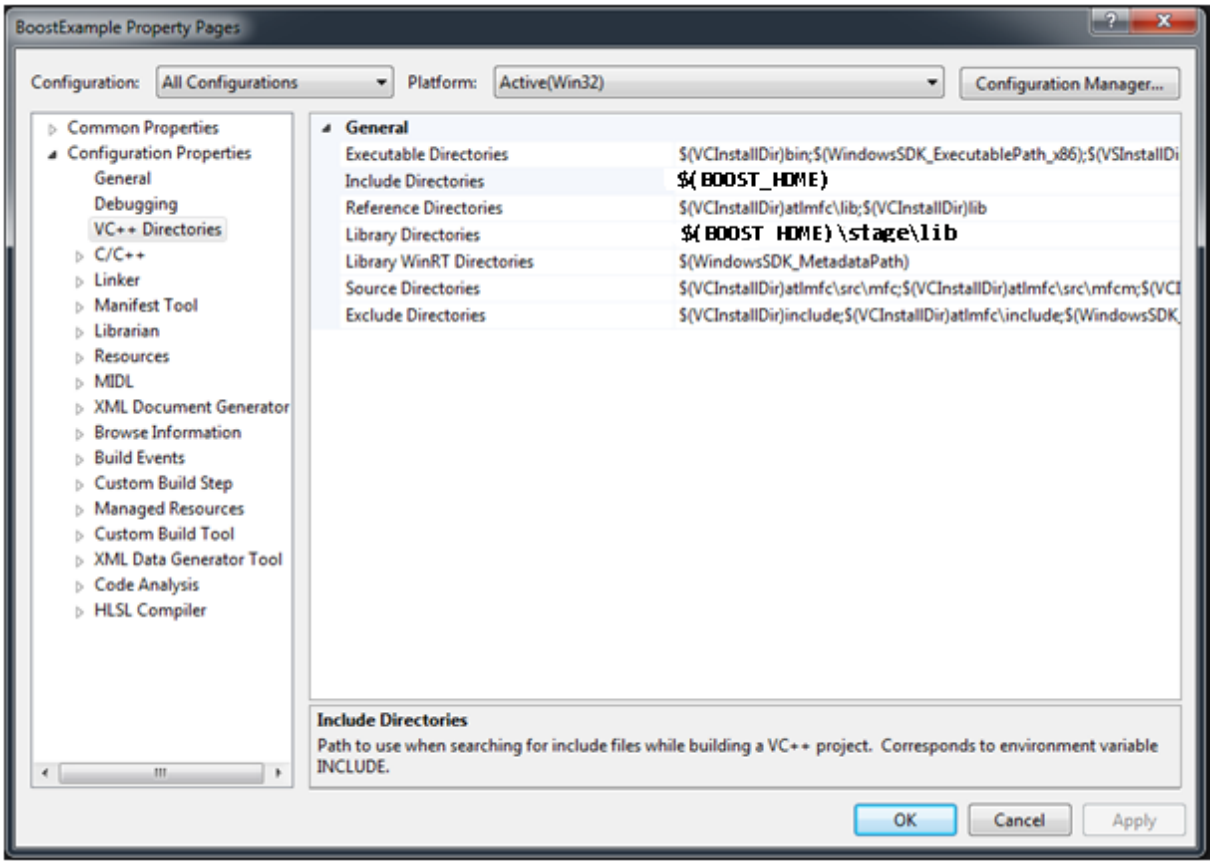


- Execute command (execution of this command can take a while):
  `"bjam toolset=msvc-11.0 --build-type=complete stage --with-thread --with-date_time"`
- After finishing the process we have built Boost libraries



Integrating VS with Boost library

- Create C++ project – empty console application
- In solution explorer click right mouse button on project name and check “Properties”
- In left top corner of window change configuration on “All Configurations”
- Check “Configuration Properties → VC++ Directories”
- Add entry “$(BOOST_HOME)” to “Include Directories” entries
- Add entry “$(BOOST_HOME)\stage\lib” to “Library Directories” entries

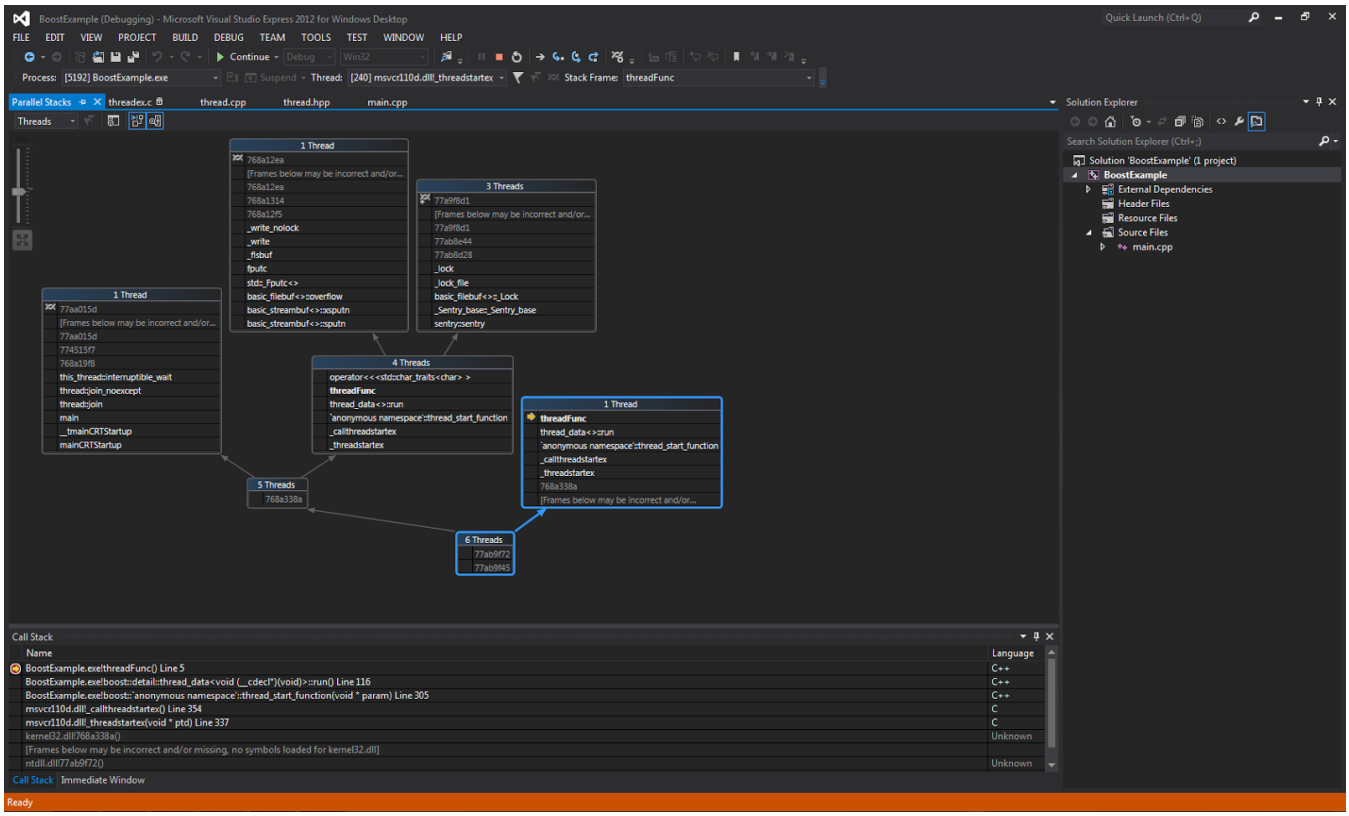


- Apply changes and exit window
- Now project supports Boost threds
- To test it, take a look at “BoostTest” project on github
  https://github.com/sw-eng-2014/CppMultithreadingOverview.git

Debugging the code

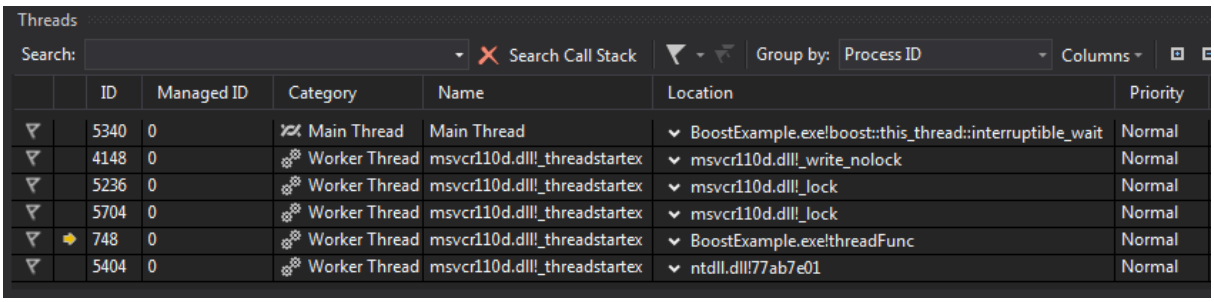
Every thread has private stack. Sometimes possibility of watching the stack is useful.
Visual Studio provide us the tool to see the stack of every thread.

- First mark the breakpoints
- Click “Debug → Start debugging”
- Click “Debug → Windows → Parallel stacks”
- The Parallel stacks windows for 6 threads looks like:



We can also trace threads using “Threads Window”

- Mark breakpoints
- Click “Debug → Start debugging”
- Click “Debug → Windows → Threads”
- Example of Threads Windows looks like:



ID	Managed ID	Category	Name	Location	Priority
5340	0	Main Thread	Main Thread	BoostExample.exe!boost::this_thread::interruptible_wait	Normal
4148	0	Worker Thread	msvcr110d.dll!_threadstartex	msvcr110d.dll!_write_nolock	Normal
5236	0	Worker Thread	msvcr110d.dll!_threadstartex	msvcr110d.dll!_lock	Normal
5704	0	Worker Thread	msvcr110d.dll!_threadstartex	msvcr110d.dll!_lock	Normal
748	0	Worker Thread	msvcr110d.dll!_threadstartex	BoostExample.exe!threadFunc	Normal
5404	0	Worker Thread	msvcr110d.dll!_threadstartex	ntdll.dll!77ab7e01	Normal

Execution of thread on indicated core

First of all we have to find the number of physical cores in our processor. To do that, we will use SYSTEM_INFO structure and GetSystemInfo procedure included in Windows.h. The following code will print number of physical cores:

```
SYSTEM_INFO sysinfo;
GetSystemInfo(&sysinfo);
printf("%d\n", sysinfo.dwNumberOfProcessors);
```

To manage threads Windows is using something called affinity mask. We distinguish 3 types of affinity masks - thread/process/system affinity mask. On 64-bit system every affinity mask is a 64-bit vector. If k'th bit of this vector is set 1, then thread, process or system is allowed to use k'th core.

From program level we can manipulate only thread and process affinity masks. To see how are set system and process affinity masks we can use the following code:

```
DWORD_PTR dwProcessAffinity, dwSystemAffinity;
bool res = GetProcessAffinityMask(GetCurrentProcess(),
&dwProcessAffinity, &dwSystemAffinity);
if(res) printf("Process:%#04x System:%#04x\n", dwProcessAffinity,
dwSystemAffinity);
```

There is the rule that if k'th bit of system's affinity mask is set 0, then k'th bit of process affinity mask can't be set to 1. Similarly when k'th bit of process's affinity mask is set to 0, then k'th bit of thread's affinity mask can't be set to 1.

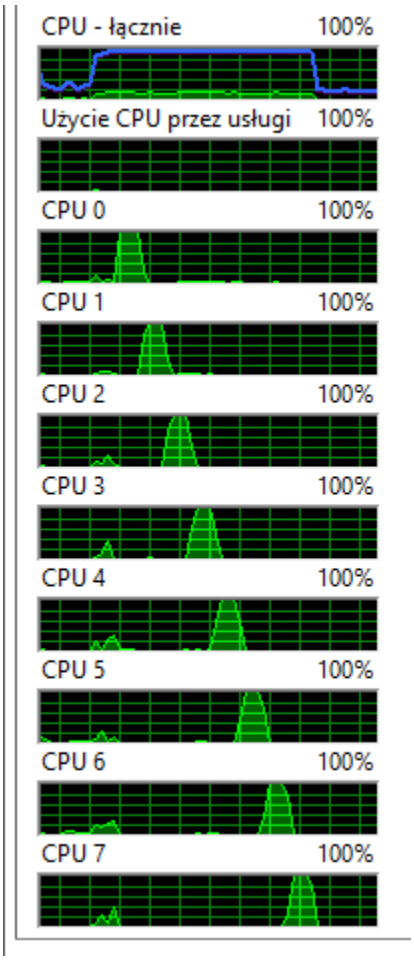
We can change process affinity mask using SetProcessAffinityMask procedure. The following code change current's process affinity mask to 0x02. After execution of this procedure current process will be allowed to run only on core number 1.

```
DWORD_PTR dwProcessAffinity = 0x02;
SetProcessAffinityMask(GetCurrentProcess(), dwProcessAffinity);
```

When we want to change thread's affinity mask, we use SetThreadAffinityMask procedure, very similiar vay to SetProcessAffinityMask. The following example allows current's thread to run only using core number 0:

```
SetThreadAffinityMask(GetCurrentThread(), 0x01);
```

In task manager we can see usage of each cores. The complete code executing task on each core one by one you can find in “AffinityTest” project on github **https://github.com/sw-eng-2014/CppMultithreadingOverview.git**.


CPU - łącznie
100%
Użycie CPU przez usługi
100%
CPU 0
100%
CPU 1
100%
CPU 2
100%
CPU 3
100%
CPU 4
100%
CPU 5
100%
CPU 6
100%
CPU 7
100%

References

Boost threads overview:
http://www.boost.org/doc/libs/1_56_0/doc/html/thread.html
Parallel stacks window overview:
http://msdn.microsoft.com/en-us/library/dd998398.aspx
Threads window overview:
http://msdn.microsoft.com/en-us/library/w15yf86f.aspx
Thread2Core:
http://msdn.microsoft.com/en-us/library/windows/desktop/ms724958(v=vs.85).aspx
http://msdn.microsoft.com/en-us/library/windows/desktop/ms683213(v=vs.85).aspx
http://msdn.microsoft.com/en-us/library/windows/desktop/ms686223(v=vs.85).aspx
http://msdn.microsoft.com/en-us/library/windows/desktop/ms686247(v=vs.85).aspx