Using distributed build system

Tamas Santa

2014

Contents

[Overview 2](#_Toc377478888)

[Incredibuild 2](#_Toc377478889)

[Test Setup 3](#_Toc377478890)

[Hardware Environment 3](#_Toc377478891)

[Physical PCs 3](#_Toc377478892)

[Virtual Machines 3](#_Toc377478893)

[Network 3](#_Toc377478894)

[Software Environment 3](#_Toc377478895)

[Installation 4](#_Toc377478896)

[Usage 8](#_Toc377478897)

[Test Results 11](#_Toc377478898)

[Reference environments 11](#_Toc377478899)

[Mixed environments 12](#_Toc377478900)

[Conclusion 13](#_Toc377478901)

[Speed 13](#_Toc377478902)

[Bottlenecks 13](#_Toc377478903)

[Cost implications 13](#_Toc377478904)

[Usability/bugs/personal experiences 13](#_Toc377478905)

# Overview

As time passes, the size of the XY solution grows almost at a linear pace. Because of that and its implications to available developer hours there is a strong need for a solution to decrease the time required to make clean, or incremental local builds. This is not a simple issue to solve however. From many possible solutions we’ll dig deep into one; the usage of distributed build systems.

The problem of build parallelization itself is really complex. Our biggest problems currently:

* the solution contains (at the time of writing this document) 337 projects, which implicates a quite complex dependency graph limiting the ability of parallelization
* the projects are mixed native C++, C++/Cli, and .NET (C#), which requires different compilers/build mechanisms/technologies

The aforementioned issues out opted most of the publicly available distributed systems, and left us with one possible candidate; Incredibuild (<http://www.incredibuild.com/>).

# Incredibuild

“IncrediBuild turns your network into a virtual supercomputer, harnessing idle CPU cycles from remote machines even while they're in use. There's no changes to source code, and absolutely no additional hardware required.”

**The main reasons we decided to give it a go:**

* easy setup (no special configuration needed, next->next->finish)
* absolutely NO change required in the solution file (or anywhere else)
* the initial cost seemed to be promising, compared to buying new high-end workstations
* complete Visual Studio integration, and C++/.NET parallelization support

The system creates a computing grid from the developers’ workstations (and/or from dedicated computers, as we’ll see later). When somebody initiates a build (from the IDE as usual), the software transparently distribute the parallelized building tasks on the network to the other build agents. It handles everything from file copies to dependency resolutions. The number of subscribed agents is limited to 4 in the trial version. Thanks to Xoreax’s support, we could use an extended trial license (with 20 agents).

# Test Setup

We decided to test the system in different environments to see which one serves our purpose best. Our IT department provided us a great collection of test hardware to experiment with.

## Hardware Environment

### Physical PCs

#### Configuration1

4 standard I7 Workstations

* Processor: Intel Core i7 860 @ 2.8Ghz
* Memory: 16Gb DDR3
* HDD: 10k RPM HDD + 120Gb SSD

#### Configuration2

5 standard I5 Workstations

* Processor: Intel Core i5-3570 @ 3.2Ghz
* Memory: 8Gb DDR3 (upgraded to 16Gb during second test phase)
* HDD: 7.2k RPM HDD + 256Gb SSD
* RAMDISK (optional)

### Virtual Machines

5 VMs in EPAM Cloud

* Processor: 2Core Xeon (1.8Ghz)
* Memory: 8Gb
* HDD: 60Gb

### Network

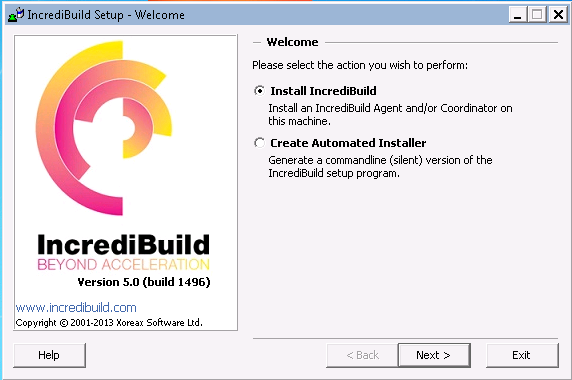
**100mbps** Ethernet Intranet  
**1Gbps** Ethernet Intranet

## Software Environment

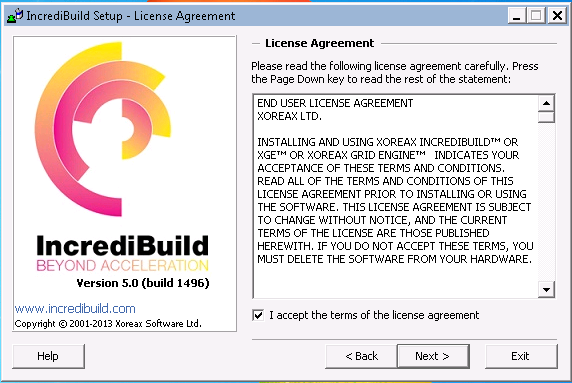
Windows 7 Enterprise  
.NET 4.5  
Visual Studio 2012  
Incredibuild 5.0 (trial)

# Installation

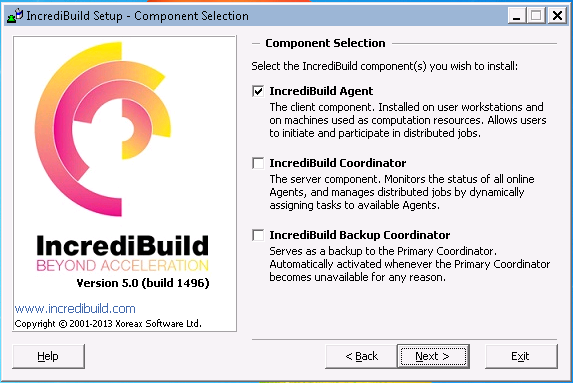
The installation is quite easy. A simple step-through process will demonstrate the installation of an agent. The coordinator needs an extra option checked. One coordinator can handle 400+ agents. An agent is capable to instantiate a build, and act as a slave at the same time.



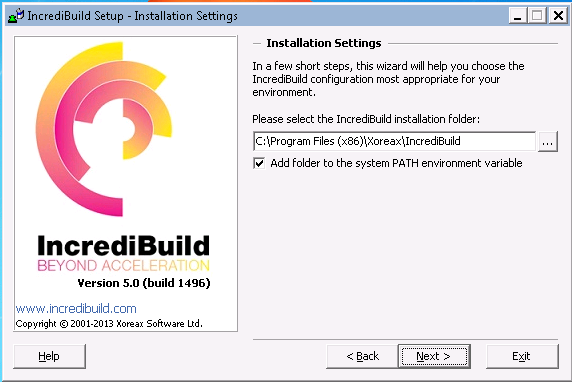
1.1 Install a separate agent, or create an automated installer



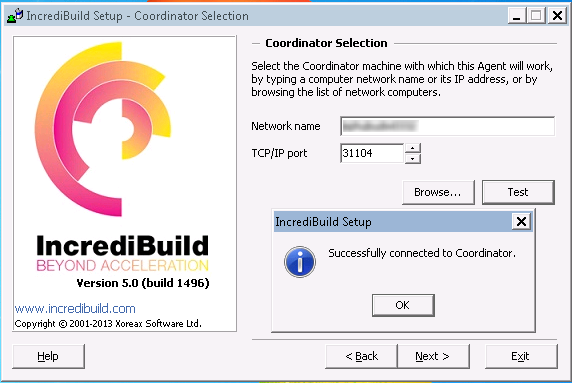
1.2 Accepting the terms



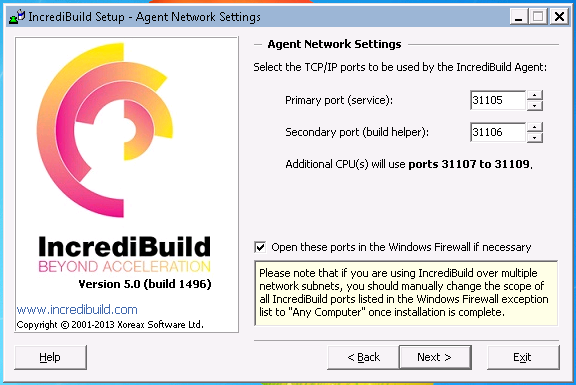
1.3 Select program components (only 1 agent needs to be a coordinator)



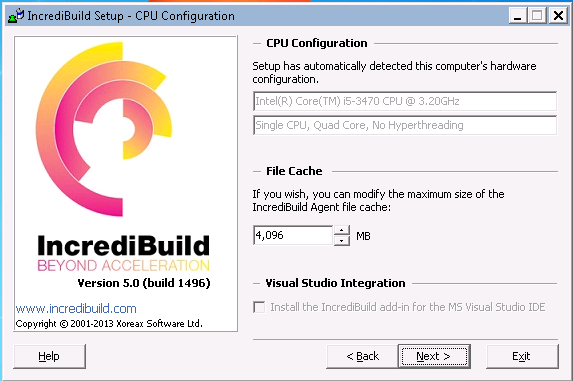
1.4 Selecting program path



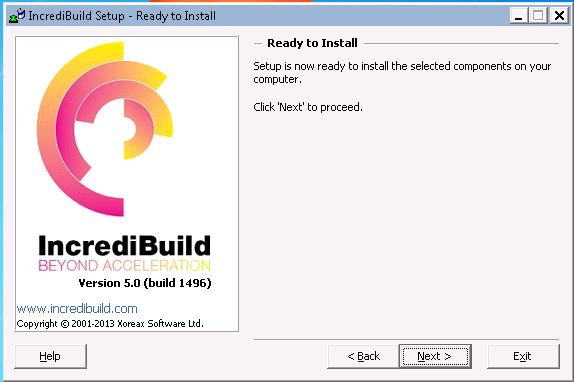
1.5 Configuring coordinator instance



1.6 Configuring network settings (leave everything on default)



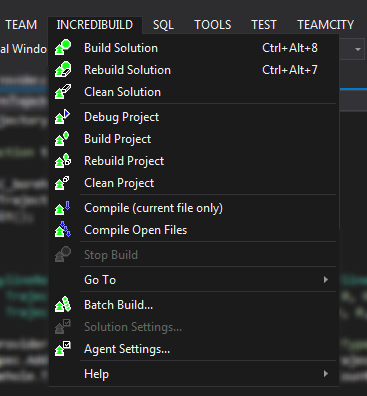
1.7 Setup file cache (8Gb is the ideal value for our project)



1.7 Everything set, proceed with installation

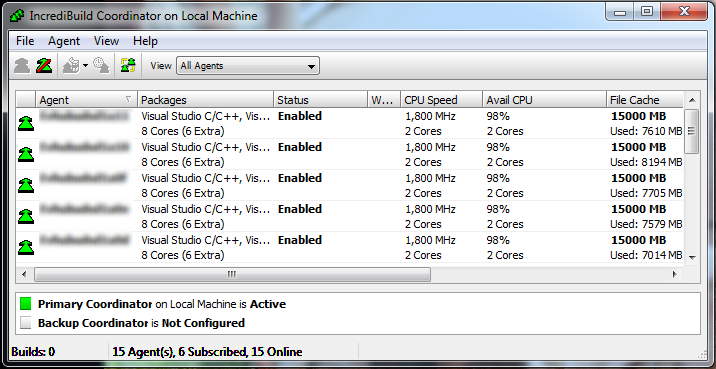
# Usage

Incredibuild does an excellent job making the distribution transparent to end-users. After installing the VS extension, we’ll have a new menu (INCREDIBUILD). From there, we can instantiate distributed builds, debugs, or other distributed functionalities. The original BUILD menu does the same as always – local builds. That way we’re able to choose the build type suits best for current needs.



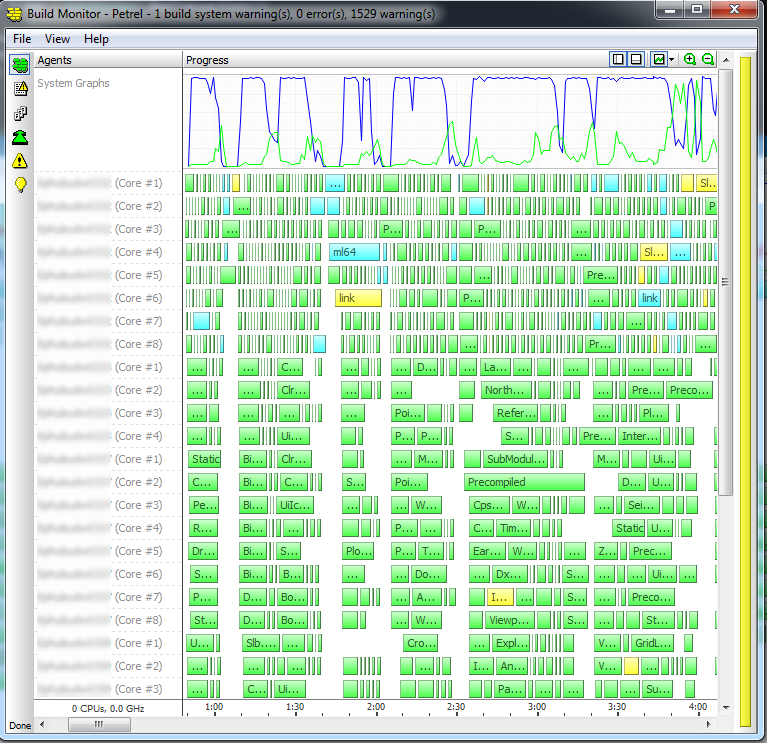
2.1 Incredibuild’s menu in Visual Studio

The coordinator instance has the ability to manage agents in the grid. This can be done via the provided management window. The remote management capabilities are great, one can configure almost everything on the agents (file cache size, firewall configuration, remote process priority, etc.).



2.2 Incredibuild Coordinator

The visual studio plugin includes a nice visualization of the distributed building process. So helpful that it even pointed out some anomalies with long linking times. A nice eye-candy for sure (better in motion).



2.3 Incredibuild build monitor

# Test Results

## Reference environments

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Configuration** | **# of cores** | **# of Agents** | **File Cache Type** | **File Cache Size** | **Build Time** | **Network** |
| 1 | I5 Workstation | 4 | 5 | Ramdisk | 3500mb | 35:54 | 100 mbps |
| I7 Workstation | 8 | 1 | SSD | 4096mb |
| 2 | I5 Workstation | 4 | 5 | Ramdisk | 8Gb | 37:23 | 100 mbps |
| I7 Workstation | 8 | 1 | SSD | 4096mb |
| 3 | I5 Workstation | 4 | 5 | Ramdisk | 3500mb | 38:46 | 100 mbps |
| 4 | I5 Workstation | 4 | 5 | Ramdisk | 8Gb | 38:54 | 100 mbps |
| 5 | I5 Workstation | 4 | 5 | HDD | 32Gb | 39:43 | 100 mbps |
| 6 | I5 Workstation | 4 | 5 | SSD | 32Gb | 40:22 | 100 mbps |
| 7 | I5 Workstation | 4 | 5 | SSD | 32Gb | 36:46 | 100 mbps |
| I7 Workstation | 8 | 1 | SSD | 4096mb |
| 8 | EPAM Cloud VM | 2 | 5 | HDD | 15Gb | 42:29 | 100 mbps |
| I7 Workstation | 8 | 1 | SSD | 4096mb |
| 9 | EPAM Cloud VM | 2 | 5 | HDD | 15Gb | 1:04:17 | 100 mbps |
| 10 | I5 Workstation | 4 | 5 | Ramdisk | 8Gb | 28:06 | gigabit |
| I7 Workstation | 8 | 1 | SSD | 4096mb |
| 11 | I5 Workstation | 4 | 5 | Ramdisk | 8Gb | 29:48 | gigabit |
| I7 Workstation | 8 | 1 | SSD | 4096mb |
| EPAM Cloud VM | 2 | 5 | HDD | 15Gb |
| 12 | I5 Workstation | 4 | 5 | Ramdisk | 8Gb | 26:06 | gigabit |
| 13 | I5 Workstation | 4 | 5 | HDD | 8Gb | 27:00 | gigabit |
| 14 | I5 Workstation | 4 | 5 | SSD | 8Gb | 26:50 | gigabit |

## Mixed environments

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Configuration** | **# of cores** | **# of Agents** | **File Cache Type** | **File Cache Size** | **Build Time** | **Network** |
| 1 | I7 Workstation | 8 | 3 | HDD | 8192mb | 35:19 | 100 mbps |
| I7 Workstation | 8 | 1 | SSD | 4096mb |
| 2 | I7 Workstation | 8 | 1 | HDD | 4096mb | 38:46 | 100 mbps |
| I7 Workstation | 8 | 1 | SSD | 4096mb |
| 3 | I7 Workstation | 8 | 3 | HDD | 8Gb | 39:27 | 100 mbps |
| I7 Workstation | 8 | 1 | SSD | 4096mb |
| I5 Workstation | 4 | 5 | HDD | 32Gb |
| EPAM Cloud VM | 2 | 5 | HDD | 15Gb |
| 4 | I7 Workstation | 8 | 2 | HDD | 8Gb | 33:04 | 100 mbps |
| I7 Workstation | 8 | 1 | SSD | 4096mb |
| I5 Workstation | 4 | 1 | SSD | 8Gb |

# Conclusion

## Speed

We were able to achieve 51% speed increase (with 5 dedicated build agents) in build time. Compared to a local build (52-60+ mins), our best measured time was 26 minutes.

We used the default recommended values, without touching our solution/project files. The only thing we changed is that half of the tests included the coordinator PC as an agent (an I7 Workstation) too. In the other half, the coordinator delegated every task to other agents (except linking, which had to happen locally). That became relevant with the gigabit connection, because in that case (as the network connection wasn’t a bottleneck anymore) the task scheduling became the main priority to the coordinator.

During build phase, we often experienced long link times (1-2mins occasionally), which cannot be parallelized. One way to solve this would be to split our project files further (and/or review the dependency graph) to make the distributed build more effective.

Please consider, that the tests were performed with ***clean*** builds. The speed improvement of an incremental build is highly dependent on the modified project, and its dependency graph.

## Bottlenecks

While the product’s info page states that it relies on heavy compression to decrease network traffic, we had to conclude that a gigabit connection is necessary to fully utilize our agents. Half of our tests were run on a 100mbps Ethernet connection, which wasn’t enough to ‘feed’ the 5 agents properly. With the gigabit connection however, we experienced a dramatic (about 20%) speed boost.

We also tested the same configurations with different disks. Ramdisk obviously performed better than the others, but there wasn’t a huge difference in build times. The 100mbps Ethernet network was bigger bottleneck than anything else. So the conclusion is that an average workstation with HDDs could perform almost as well as ones with SSDs, or Ramdisks in a real-life scenario.

## Cost implications

The licensing part of Incredibuild is a bit tricky. Let’s see an example of the required licenses for an agent with 8 cores, which is capable to initiate both native, and managed builds:

|  |  |  |  |
| --- | --- | --- | --- |
| **License Type** | **Description** | **Price** | **Period** |
| IB208 | 8 core agent license | 495$ | *one time* |
| IB301 | VS C++ solution | 99$ | *12months rental* |
| IB3010 | C# solution | 75$ | *12months rental* |
| **Sum** | | **669$** |  |

## Usability/bugs/personal experiences

The distributed build result (files and folder structure) is 100% equivalent to the local build’s output. The setup and usage is very user-friendly. Besides from some minor firewall misconfiguration *(which was kindly pointed out by the software)*, there weren’t any issues whatsoever. The trial version allows the subscription of 4 build agents, for a 30days period.

On the 2nd day of the trial, I was contacted by Xoreax’s support. They were kind enough to provide us an extended (20PCs) trial version. During our calls and emails, they provided excellent support, answered our questions in time, and gave us excellent user experience.