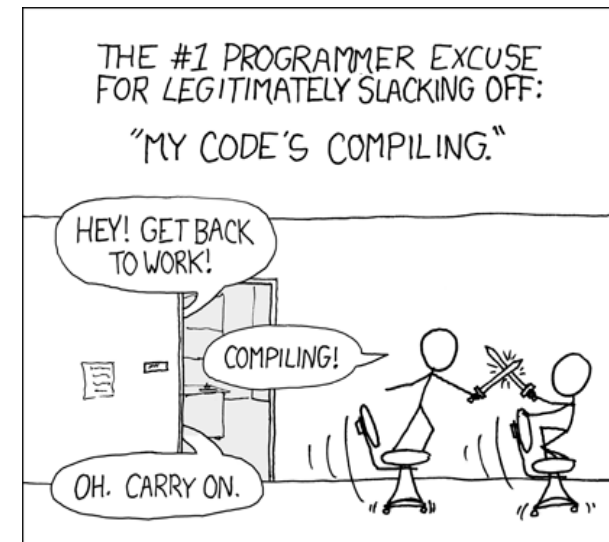


# Distributed C/C++ Compilation service

Ian Richard BAKER  
KELEMEN Péter  
CERN IT OIS

- Scope: C/C++ codebases (gcc)
- **Isolated** developer groups
- **Large** software packages
- **Long** compilation times
- **Limited** hardware resources



- Compile in **parallel**
- Throw **more CPU** at it
- **Distribute** over the network
- **Consolidate** resources
- **Scale** horizontally
- **Delegate** management
- **All** of the above: using **distcc**

- distcc is *de facto* standard
- ATLAS
  - Nightly builds using dedicated distcc cluster (lxbuild)
- CMS
  - distcc integrated with SCRAM (CHEP'04)
- LHCb
  - SLC4/SLC5 build environments, easy target
- Linux.Support (planned)
  - ...as the distribution grows, the need arises to reduce compile times of RPMs (software fixes, security updates)

- Write code (\*.c, \*.cpp)
  - Left as an exercise to the reader
- Preprocessing (\*.i)
  - cpp(1) expands #include's and macros, needs header files
- Compiling (\*.s)
  - C code is translated into assembly code
- Assembling (\*.o)
  - Executable machine code is generated from assembly source
- Linking
  - Multiple objects are stitched together into an executable, needs libraries

- Write code (\*.c, \*.cpp)

- Left as an exercise to the reader

- Preprocessing (\*.i)

- cpp(1) expands #include's and macros, needs header files

- Compiling (\*.s)

- C code is translated into assembly code

- Assembling (\*.o)

- Executable machine code is generated from assembly source

- Linking

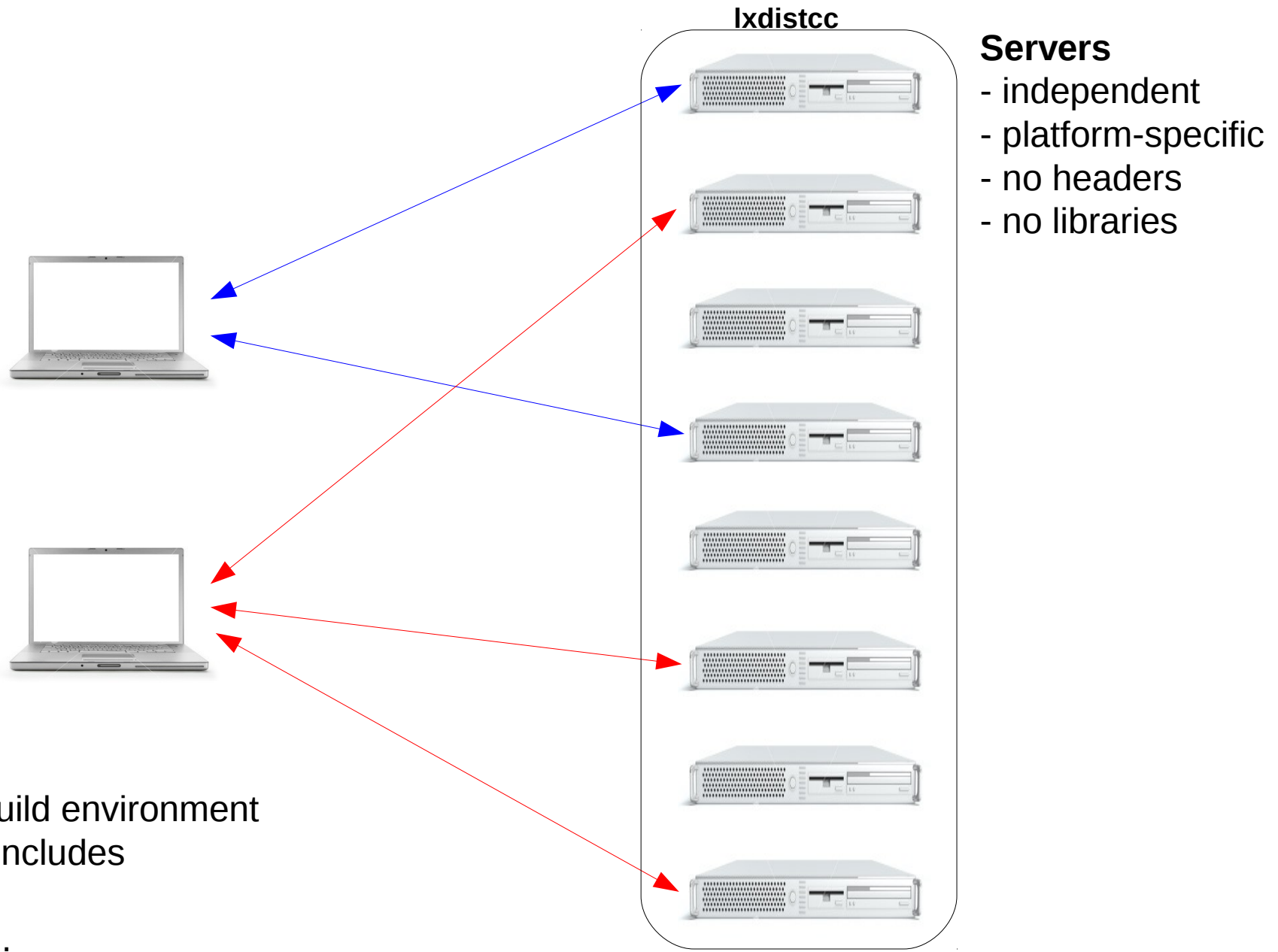
- Multiple objects are stitched together into an executable, needs libraries

The logo for distcc, featuring the word "distcc" in a stylized, 3D font with a color gradient from red to yellow.

- **Compilation unit**
  - Preprocessed C/C++ source ready to be compiled.
- **Object file**
  - Linkable binary code, result of the compilation.
- **Client**
  - Where your software build is running.
- **Server(s)**
  - Where the compilation jobs are dispatched to.

- Wrapper around gcc(1)
- client/server model
  - One client, many servers talking via a network protocol
  - No shared filesystems required
  - No virtual machines required
- client: preprocessing
  - Using your regular build environment (include headers)
- server: compilation
  - Self-contained compilation units, no need for headers/libraries
- client: linking
  - Using your regular build environment (shared libraries)





- **Compiler versions**

- distcc is compiler-agnostic
- client/server versions are *strongly advised* to match
- Technically it is possible to support multiple compilers

- **Platforms**

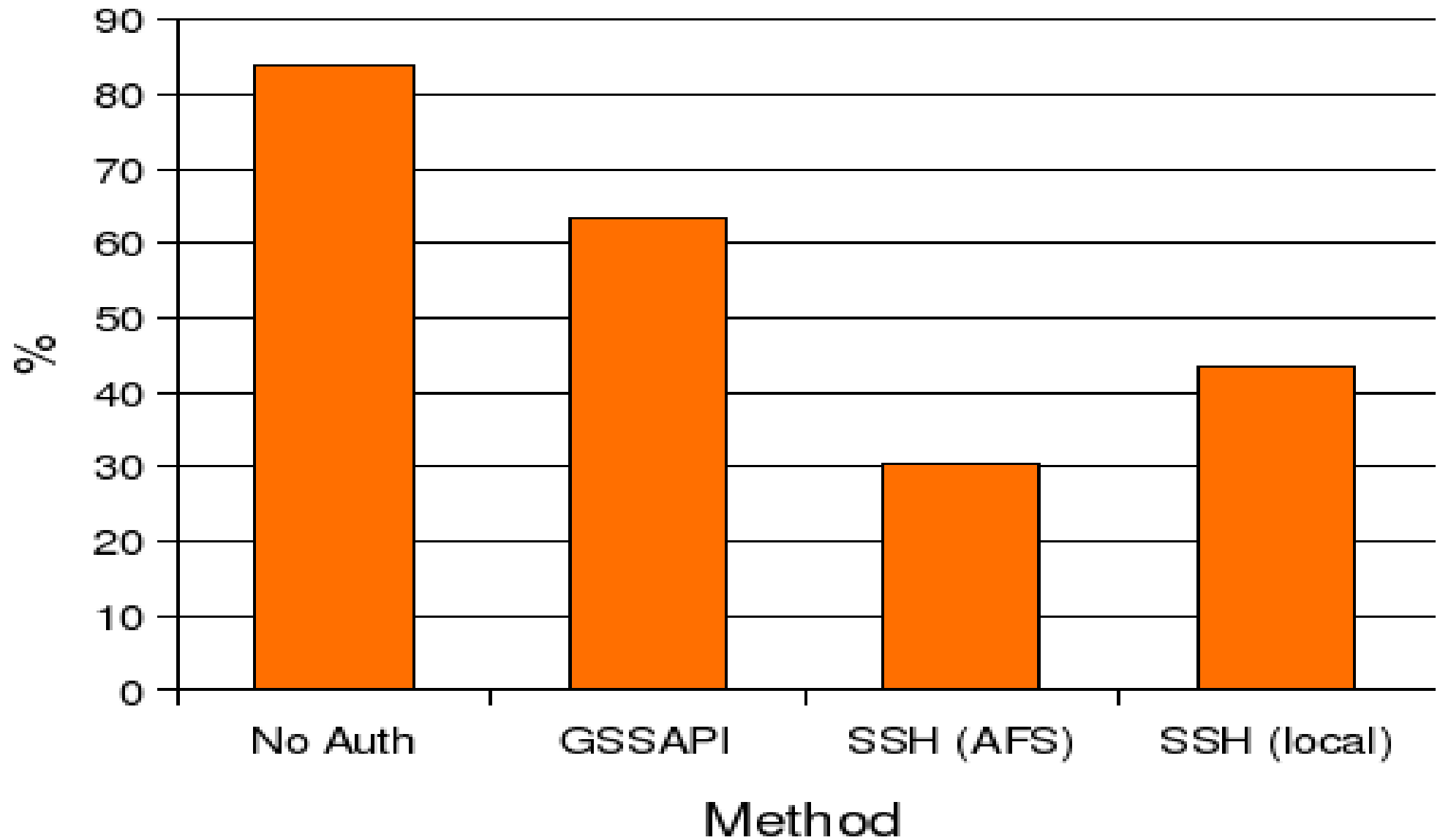
- Linking objects compiled on different platforms will obviously fail
- Best practice is to have per-platform capacity allocated
- Possibility to grow/shrink capacity just like lxbatch

- **You: parallelizable SW build**

- The fewer compilation ordering dependencies, the better
- If **make -j** fails locally, your project cannot build in parallel
- If **make -j** is not faster on multiple CPUs, distcc won't help either

- **GSSAPI authentication**
  - Prerequisite for a shared resource
  - Currently Kerberos V
- **whitelist / blacklist**
- **Log timestamps**
- **...submitted to upstream**
  - Google is current maintainer
- **Client integrated in SLC4/SLC5**
- **User prioritization (planned)**

## Comparison of Efficiency



- **Quattorized server nodes**
  - 8x x86\_64 E5410 cores SLC4/64-bit
  - 8x x86\_64 E5410 cores SLC5/64-bit
  - Easy to add more machines (and architectures)
- **Client RPM in SLC4/SLC5**
  - Available off-the-shelf
  - gcc4.3 and LCG gcc versions available with SLC5
- **Shared service**
  - All LHC experiments
  - No registration required, but we'd like to hear from you!
- **Authentication (GSSAPI)**
  - Users should present valid Kerberos 5 principal

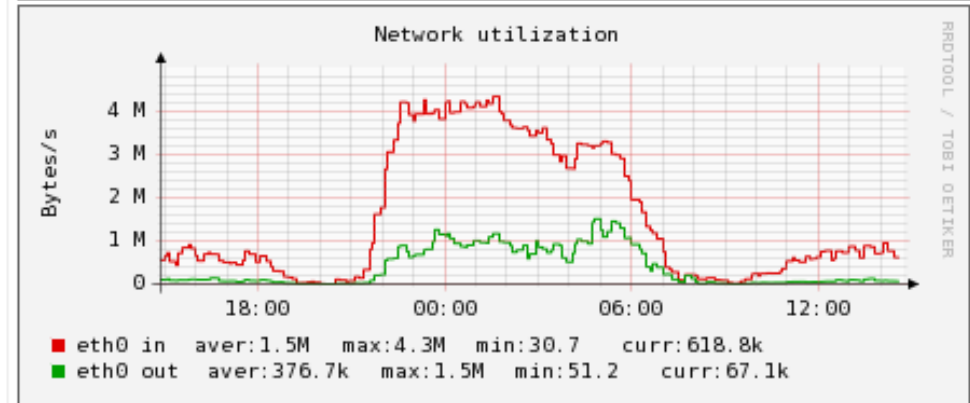
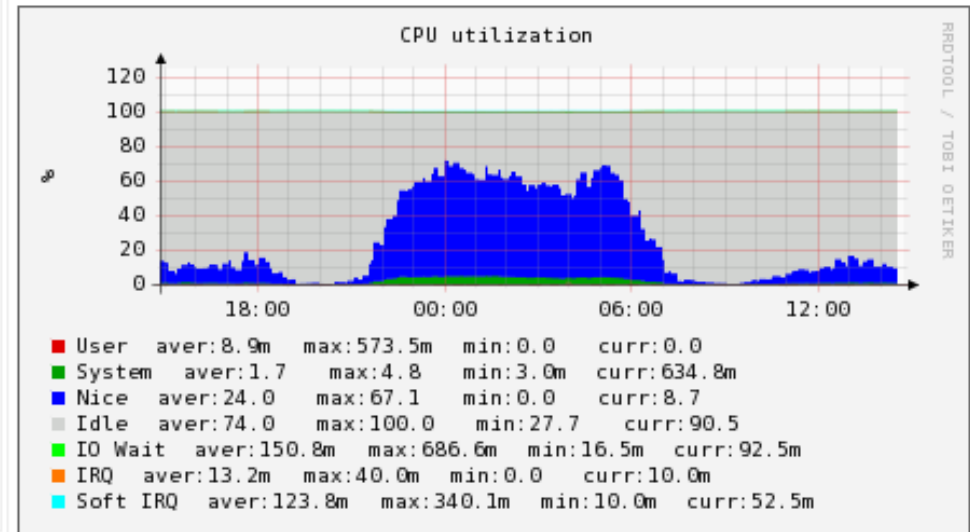
- 2008Q2: project start
- 2008Q3: securing hardware
- 2008Q4: GSSAPI impl'd
- 2009Q1: pilot opens
  - 64 cores w/ SLC4/system compiler
  - 64 cores w/ SLC5/system compiler
- 2009Q3: established service
  - LCG compilers available on cluster
  - Statistics, REMEDY support line, LEMON monitoring, ...
- 2010Q4: 1yr service (10% FTE)

- [Home](#)
- [Documentation](#)
- [Alarms](#)
- [Metrics](#)
- [Misc](#)
- [Help](#)

## Information for Clusters / Ixdistcc / Ixbrb0601

### Host information

<b>operating system(s)</b>	Scientific Linux CERN SLC release 5.4 (Boron)
<b>achitecture (kernel)</b>	x86_64 (2.6.18-164.6.1.el5)
<b>up time (since)</b>	337 days, 7h:19m (Wed, 02 Dec 2009 07:31:21+0100)
<b>CPU (count/logical)</b>	Intel(R) Xeon(R) CPU E5410 @ 2.33GHz (2/8)
<b>memory (swap)</b>	16058 MB (4095 MB)
<b>cluster (subcluster)</b>	<b>Ixdistcc (Ixdistcc/x86_64_slc5)</b>
<b>IP address(es)</b>	128.142.195.10 (eth0)
<b>state</b>	production
<b>status</b>	Available



Set span:  offset:

Search entities:  Virtual Clusters ▾ Clusters ▾ Racks ▾ Hardware types ▾ Services ▾

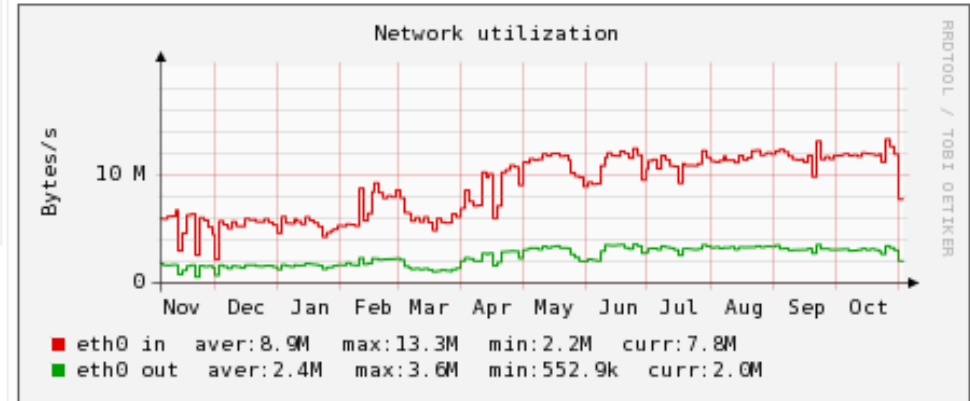
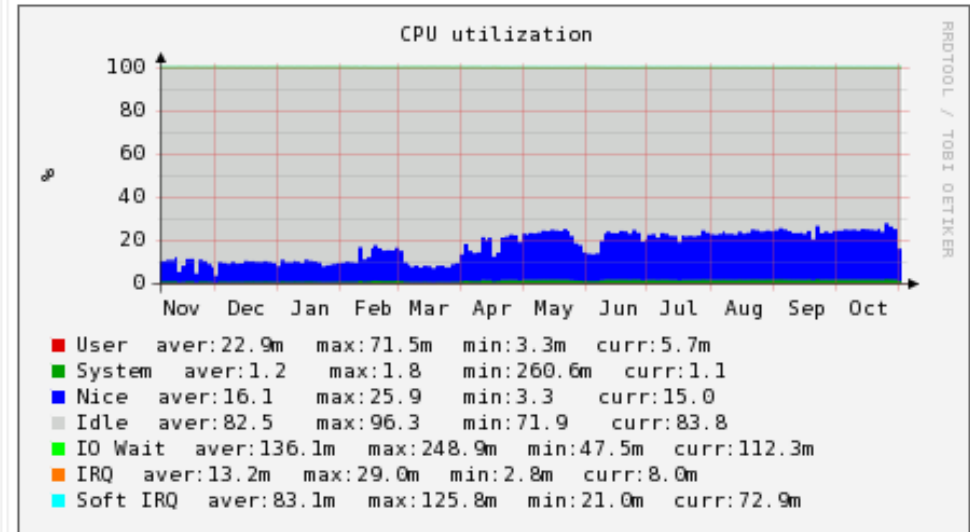
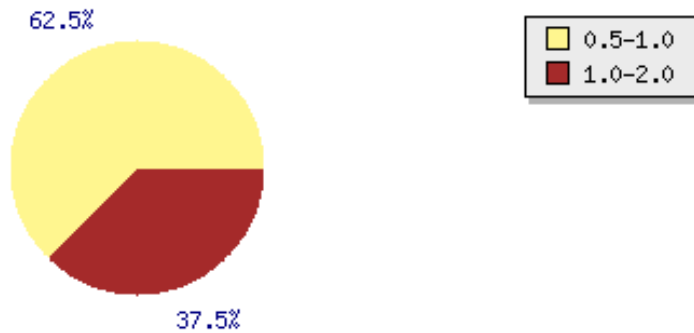
- [Home](#)
- [Documentation](#)
- [Alarms](#)
- [Metrics](#)
- [Misc](#)
- [Help](#)

## Information for Clusters / Ixdistcc / Ixdistcc/x86\_64\_slc5

### Cluster information

<b>number of hosts (down)</b>	8 (0)
<b>operating system(s)</b>	Scientific Linux CERN SLC release 5.4 (Boron)
<b>average of up times</b>	<b>301 days, 6h:24m</b>
<b>select from hosts</b>	None selected ▼

Load average distribution

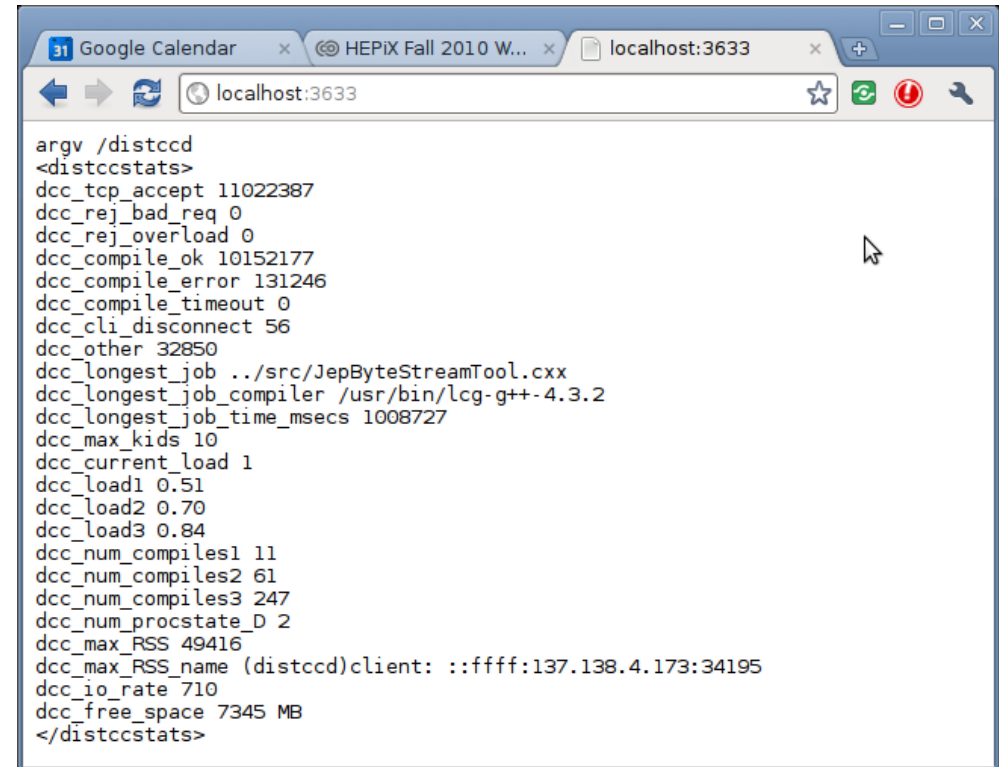


Set span:  offset:

Search entities:  Virtual Clusters ▶ Clusters ▶ Racks ▶ Hardware types ▶ Services ▶



- ~10M/337 days
- ~30k / day
- ~20 / minute
  - Longest job takes ~16 mins

A screenshot of a web browser window displaying distcc statistics. The browser has three tabs: 'Google Calendar', 'HEPIX Fall 2010 W...', and 'localhost:3633'. The address bar shows 'localhost:3633'. The main content area displays the output of a distcc client command, showing various statistics such as TCP accepts, compile errors, and current load.

```
argv /distccd
<distccstats>
dcc_tcp_accept 11022387
dcc_rej_bad_req 0
dcc_rej_overload 0
dcc_compile_ok 10152177
dcc_compile_error 131246
dcc_compile_timeout 0
dcc_cli_disconnect 56
dcc_other 32850
dcc_longest_job ../src/JepByteStreamTool.cxx
dcc_longest_job_compiler /usr/bin/lcg-g++-4.3.2
dcc_longest_job_time_msecs 1008727
dcc_max_kids 10
dcc_current_load 1
dcc_load1 0.51
dcc_load2 0.70
dcc_load3 0.84
dcc_num_compiles1 11
dcc_num_compiles2 61
dcc_num_compiles3 247
dcc_num_procstate_D 2
dcc_max_RSS 49416
dcc_max_RSS_name (distccd)client: ::ffff:137.138.4.173:34195
dcc_io_rate 710
dcc_free_space 7345 MB
</distccstats>
```

- Intermittent auth timeouts
  - Currently investigating...
- Move to virtual machines
  - Recent tests of almost-bare-metal CPU performance is promising
- Log analysis for statistics
  - Move to rsyslog from dedicated logfile
  - Later centralize logging
- Introduce pump-mode
  - Requires porting work for the python include-server for SLC5
- Look at Google's next-gen project
  - Whenever it becomes open-source... principle is constant recompilation based on source dependencies

- Write code (\*.c, \*.cpp)

- Left as an exercise to the reader

- Preprocessing (\*.i)

- cpp(1) expands #include's and macros, needs header files

- Compiling (\*.s)

- C code is translated into assembly code

- Assembling (\*.o)

- Executable machine code is generated from assembly source

- Linking

- Multiple objects are stitched together into an executable, needs libraries

Distcc  
pump

- Invite feedback, suggestions
  - [linux-distcc@cern.ch](mailto:linux-distcc@cern.ch), open mailing list
  - User documentation
- Work with early adopters
  - Migrate existing distcc users (DONE)
  - Invite new interested parties
- Spread the word!
- User docs in Wiki

<https://twiki.cern.ch/twiki/bin/view/LinuxSupport/DistccPilotService>

Thank you for your attention.