Distributed C/C++ compilation service pilot

Ian Richard BAKER
KELEMEN Péter
CERN IT FIO

Problem

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

Solution

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

Why now?

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

Compilation

- 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

Compilation (distcc)

- 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

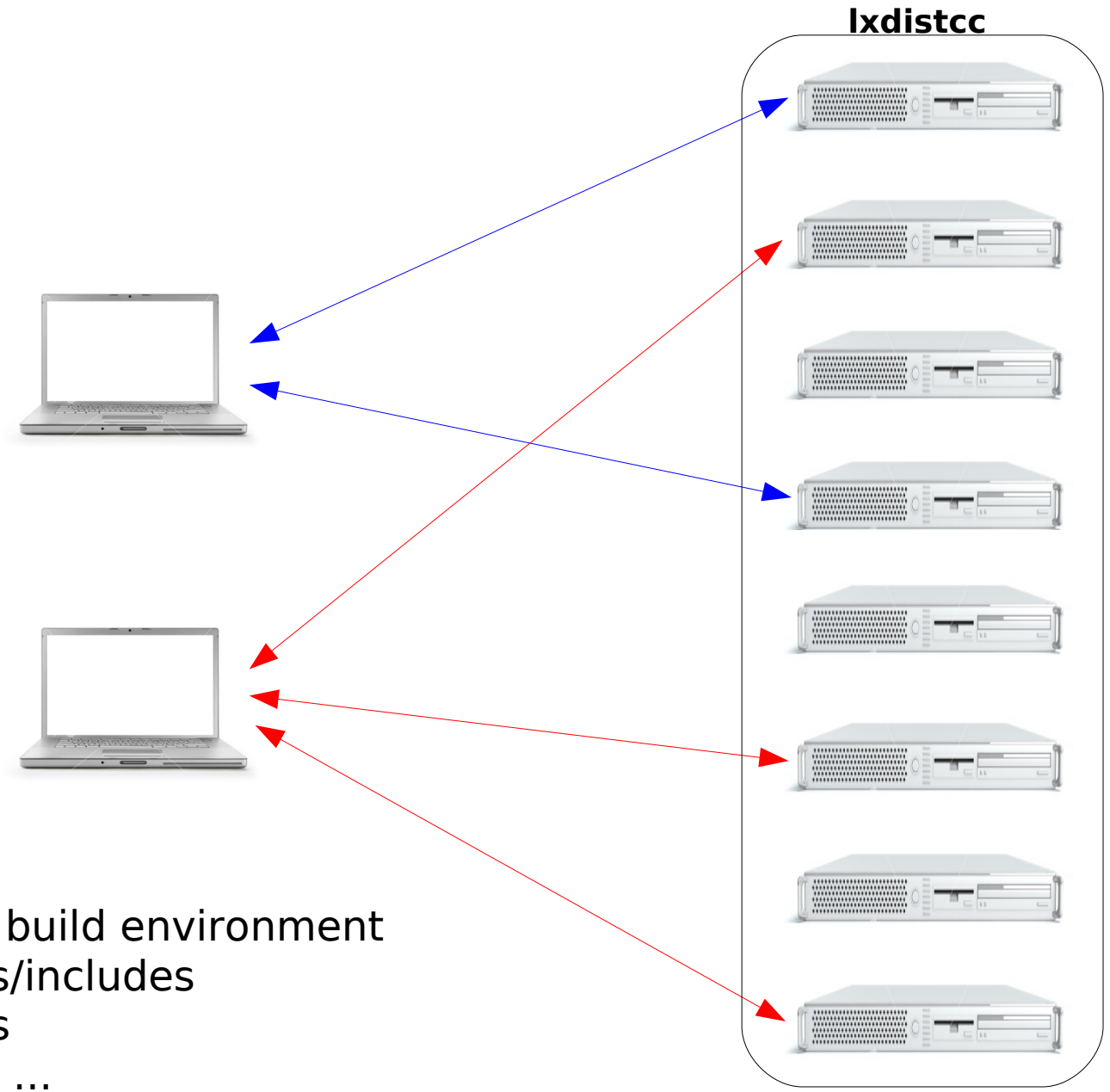
Terminology

- 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.

Technology

- 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)

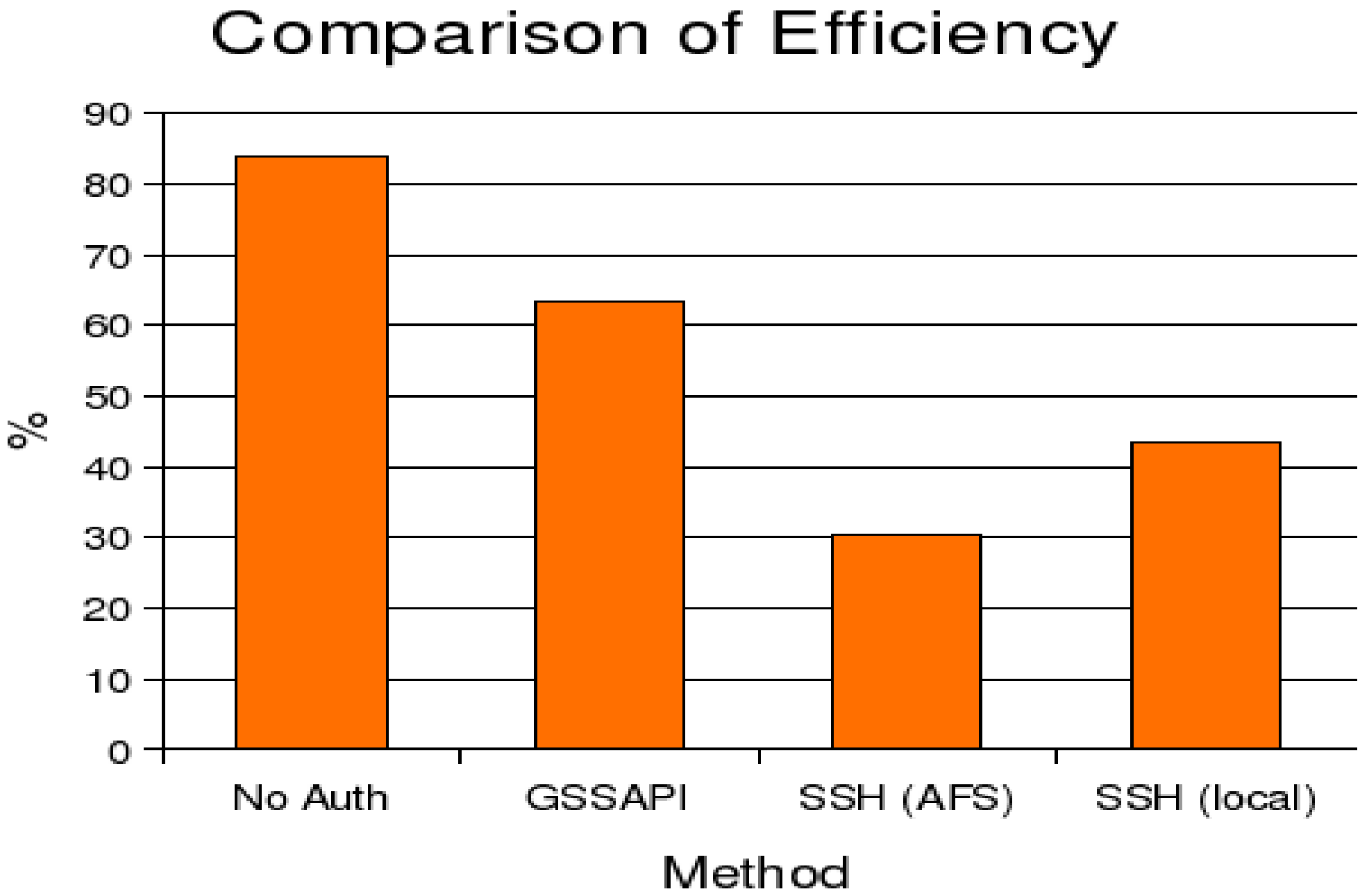
Technology



Servers
- independent
- platform-specific
- no headers
- no libraries

Clients
- regular build environment
- headers/includes
- libraries
- editors, ...

Linux kernel build



No Silver Bullet Though...

- 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

CERN Improvements to distcc

- GSSAPI authentication
  - Prerequisite for a shared resource
  - Currently Kerberos V
- whitelist / blacklist
- Log timestamps
- …submitted to upstream
  - Google is current maintainer
- Client integrated in SLC4/5
- User prioritization (planned)

CERN IT lxdistcc Pilot

- Quattorized server nodes
  - 8x8=64 E5410 cores SLC4/64-bit (ready)
  - 8x8=64 E5410 cores SLC5/64-bit (hardware allocated)
  - Easy to add more machines (and architectures)
- Client RPM in SLC4
  - SLC5 is coming Real Soon Now™
  - gcc4.3 is foreseen to be 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

Timeline

- 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, gcc4.3 by mid-March
- 2009Q3: established service
  - Depending on demand and the success of the pilot
  - Statistics, REMEDY support line, LEMON monitoring, ...

Build a Community

- Invite feedback, suggestions
  - linux-distcc@cern.ch, open mailing list
  - User documentation
- Work with early adopters
  - Migrate existing distcc users
  - Invite new interested parties
- User docs in Wiki

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

Questions?

Thank you for your attention.

Compilation (pump)

- 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