Faster
Software Builds

John Ousterhout

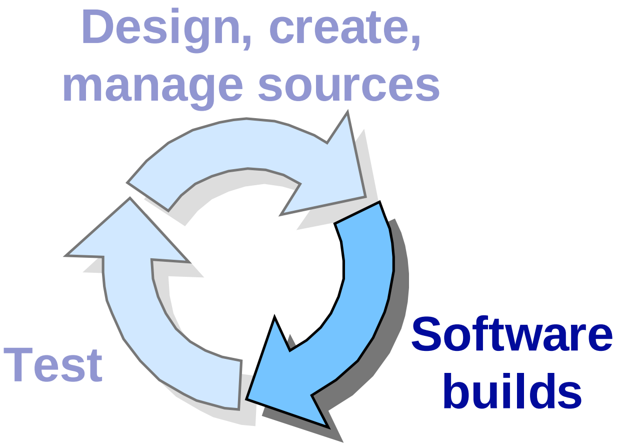
electric cloud

2307 Leghorn Street
Mountain View, CA 94043
www.electric-cloud.com

Overview

electric cloud

- **Slow builds impact almost all medium/large development teams**
- **Electric Cloud speeds up builds 10-20x:**
  - Harnesses clusters of inexpensive servers
  - Unlocks concurrency by deducing dependencies
  - Minimizes scalability bottlenecks
- **Faster builds mean**
  - Faster time to market
  - Higher product quality
  - Ability to do more with less



Outline

electric cloud

- The impact of slow builds
- The holy grail: concurrent builds
- Dependencies: problem and solution
- Electric Cloud architecture
- Managing files
- Limiting bottlenecks
- Performance measurements

Problem: Slow Builds

electric cloud

Over 500 companies surveyed, average build 2-4 hours

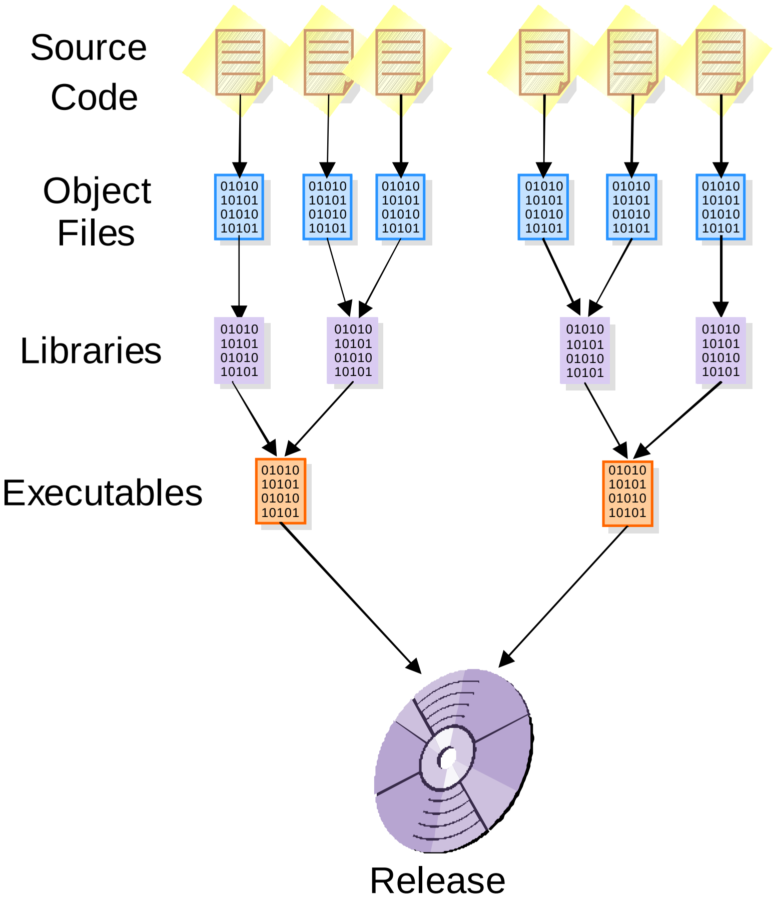
- **5-15% loss in engineering productivity:**
  - Wasted engineering time & frustration
  - Less time to fix bugs, add features
- **5-10% delay in time to market:**
  - Slow builds add weeks to release cycles
  - Uncertainty & risk due to last-minute broken builds
- **Quality & customer satisfaction:**
  - Developers can't rebuild before check-in
  - QA waiting on broken builds or skipping tests to meet deadlines
  - More bugs escape to the field

Personal Experience

electric cloud

- **Slow builds drove me crazy**
  - Sprite research project (Berkeley, late ’80s):
    - Most popular feature was “pmake”
    - Painful to return to traditional OS
  - Interwoven, 2000-2001:
    - 7-10-hour builds
    - > 1 month with no successful daily builds, late in a release cycle
- **Discovered that they drive everyone crazy!**
- **Founded Electric Cloud to solve the problem**

Theoretical Solution: Concurrency



Source Code

Object Files

Libraries

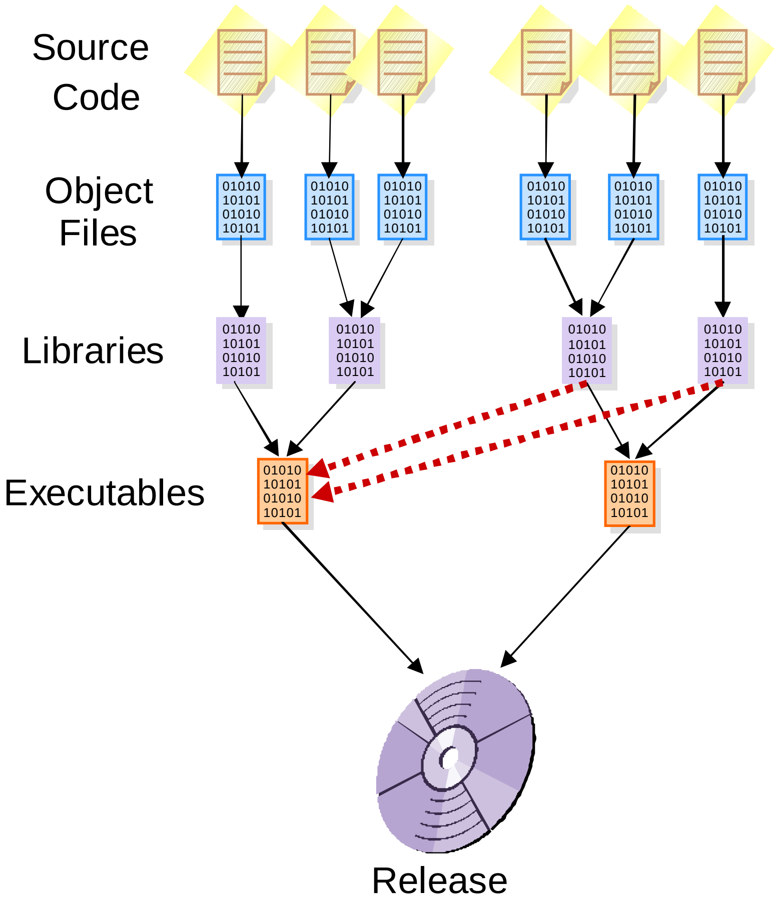
Executables

Release

- **Builds have inherent parallelism**
- **Solution: split up builds and run pieces concurrently**
  - Large SMP Machines (gmake –j)
  - Distributed builds (distcc)

If only it were this easy…

Problem: Dependencies



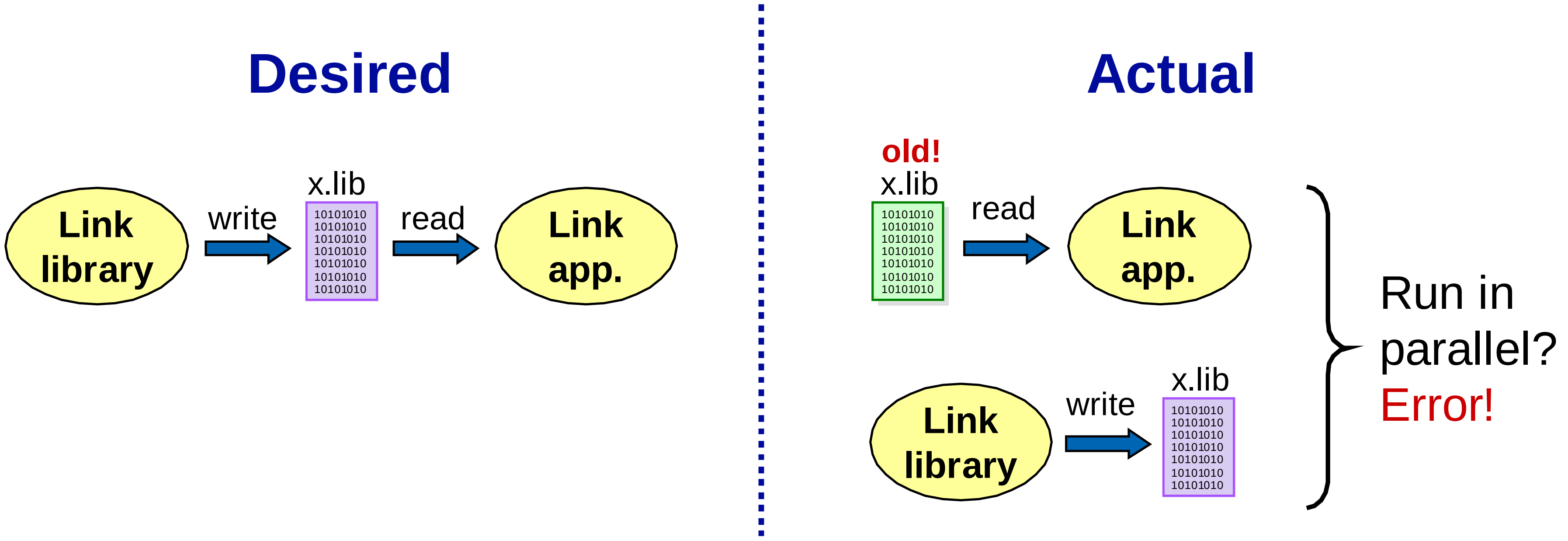
- Builds have inherent parallelism
- Solution: split up builds and run pieces concurrently
  - Large SMP Machines (gmake –j)
  - Distributed builds (distcc)
- **Current attempts to speed builds yield small results**
- **Dependency problems:**
  - Incomplete
  - Can’t be expressed between Makefiles
  - Result: broken builds

Difficult to get more than a 2-3x speedup

Hard to maintain Makefiles

Electric Cloud Solution

- **Deduce dependencies on-the-fly:**
  - Watch all file accesses: these indicate dependencies
  - Automatically detect out-of-order steps



Desired

Link library → write → x.lib → read → Link app.

Actual

old! x.lib → read → Link app.

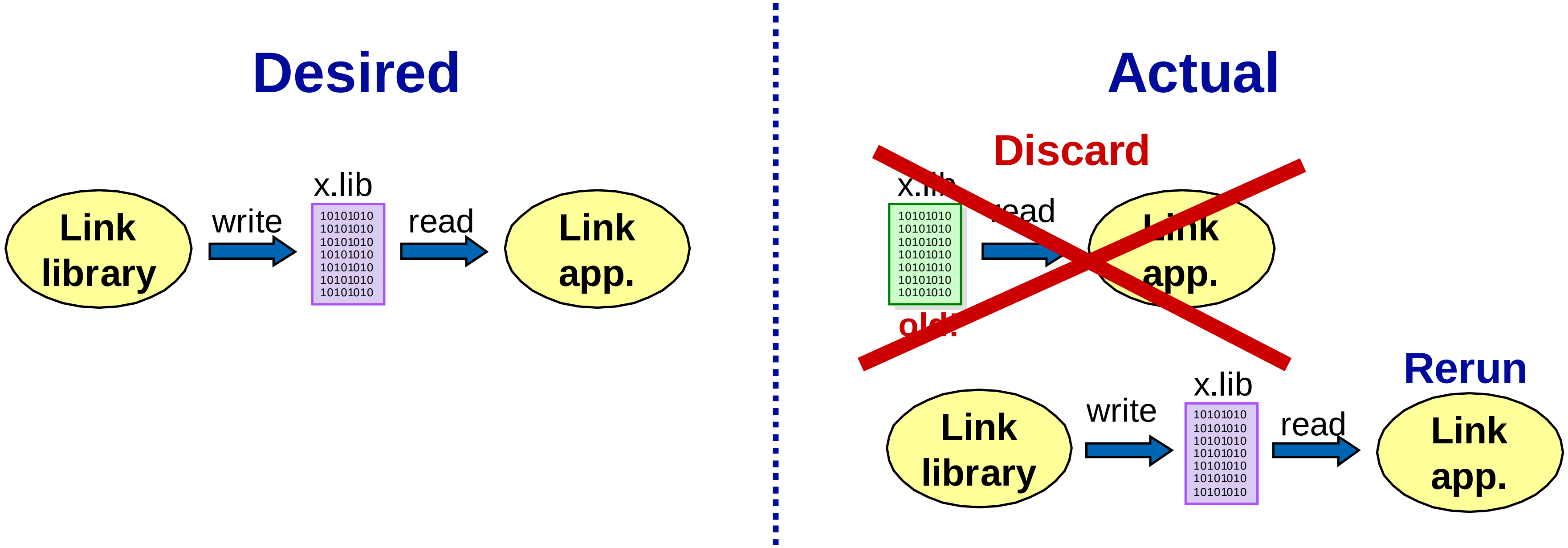
Link library → write → x.lib

Run in parallel? Error!

Electric Cloud Solution

electric cloud

- **Deduce dependencies on-the-fly:**
  - Watch all file accesses: these indicate dependencies
  - Automatically detect and correct out-of-order steps
  - Save discovered dependencies for future builds
  - Result: high concurrency possible



Desired

Link library → write → x.lib → read → Link app.

10101010
10101010
10101010
10101010
10101010
10101010
10101010

Actual

Discard

x.lib → read → Link app.

old!

10101010
10101010
10101010
10101010
10101010
10101010
10101010

Rerun

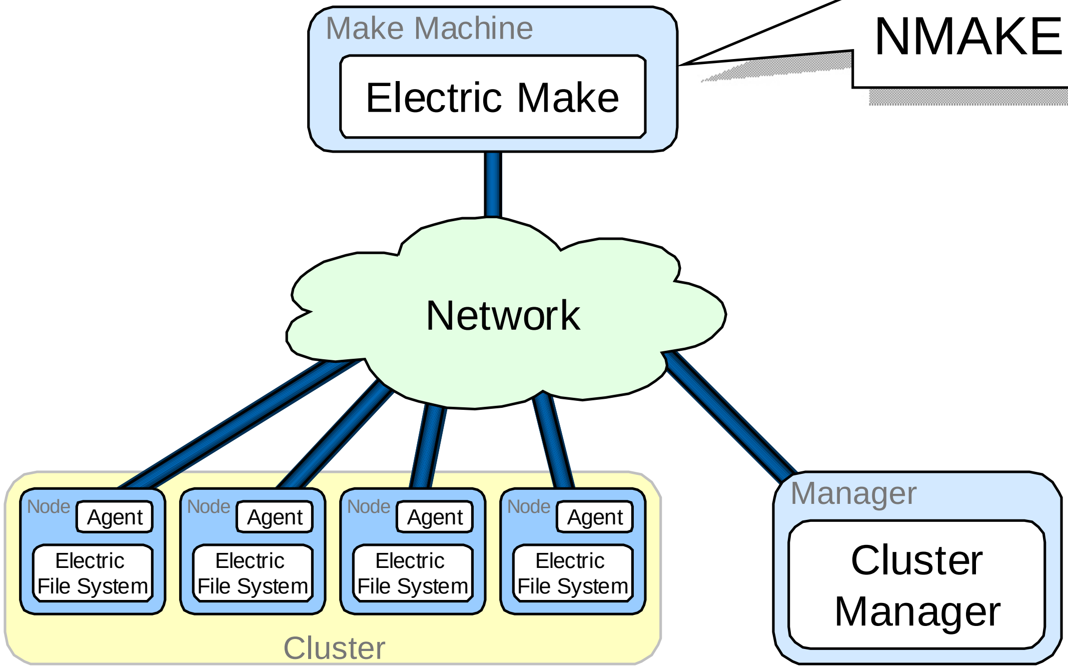
Link library → write → x.lib → read → Link app.

10101010
10101010
10101010
10101010
10101010
10101010
10101010

Electric Cloud Architecture

electric cloud

Plug-in replacement for GNU Make, Microsoft NMAKE



Inexpensive rack-mounted servers run pieces of build in parallel

Web-based reporting, management tools

Clustering Approach

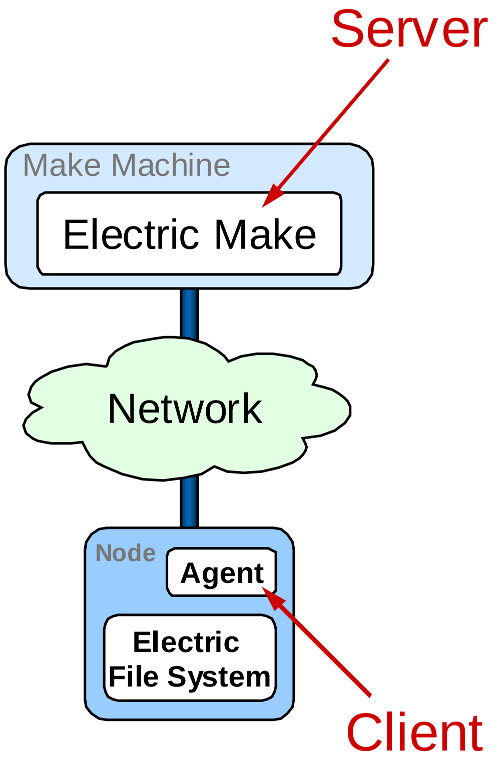
electric cloud

- **Advantages (vs. multiprocessor):**
  - Cost-effective: $1-2K per CPU
  - Scalable: no hard limit to cluster size

- **Potential problems:**
  - Build state not necessarily available on nodes
  - Overhead for network communication
  - Robustness: more pieces that can break

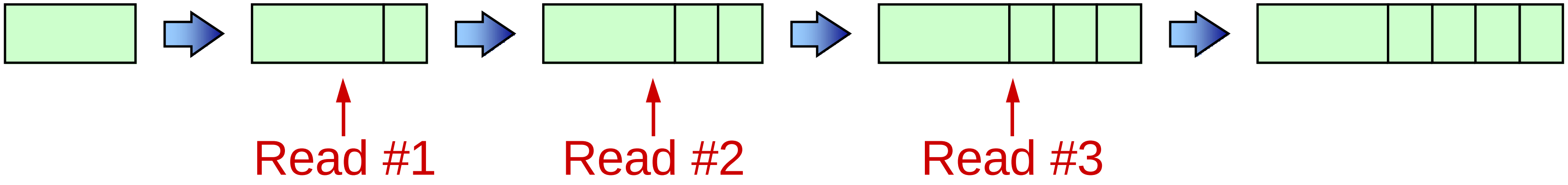
Virtualization

- **Node environment must duplicate make machine; hard because of**
  - ClearCase views
  - Different environments on different make machines
  - File versioning within a build
- **Simple application-specific network file system:**
  - Electric Make is server
  - Agent is client, fetches files on demand
  - Virtualizes subtree(s) from make machine
  - Files cached on nodes during a build
- **On Windows, registry data is also virtualized on nodes**



Versioning File System

Example: log file extended with series of appends



Read #1

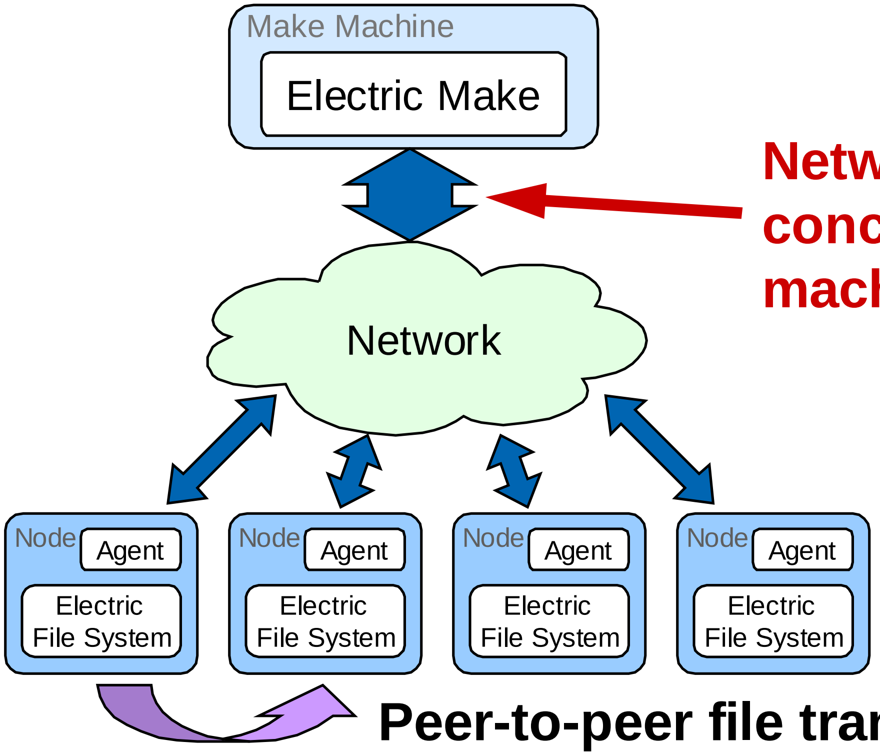
Read #2

Read #3

- **Files can have many versions during build:**
  - Append to log file
  - Debug/release versions compiled to same .o files
- **Each read must return correct version (based on *sequential order* for build)**
- **Electric Make maintains version history for each file**
  - Tricky: name space must be versioned also
- **Network file system passes appropriate version to each job, flushes caches when necessary**

Network Optimization

electric cloud



Network bandwidth concentrates at make machine

Peer-to-peer file transfer

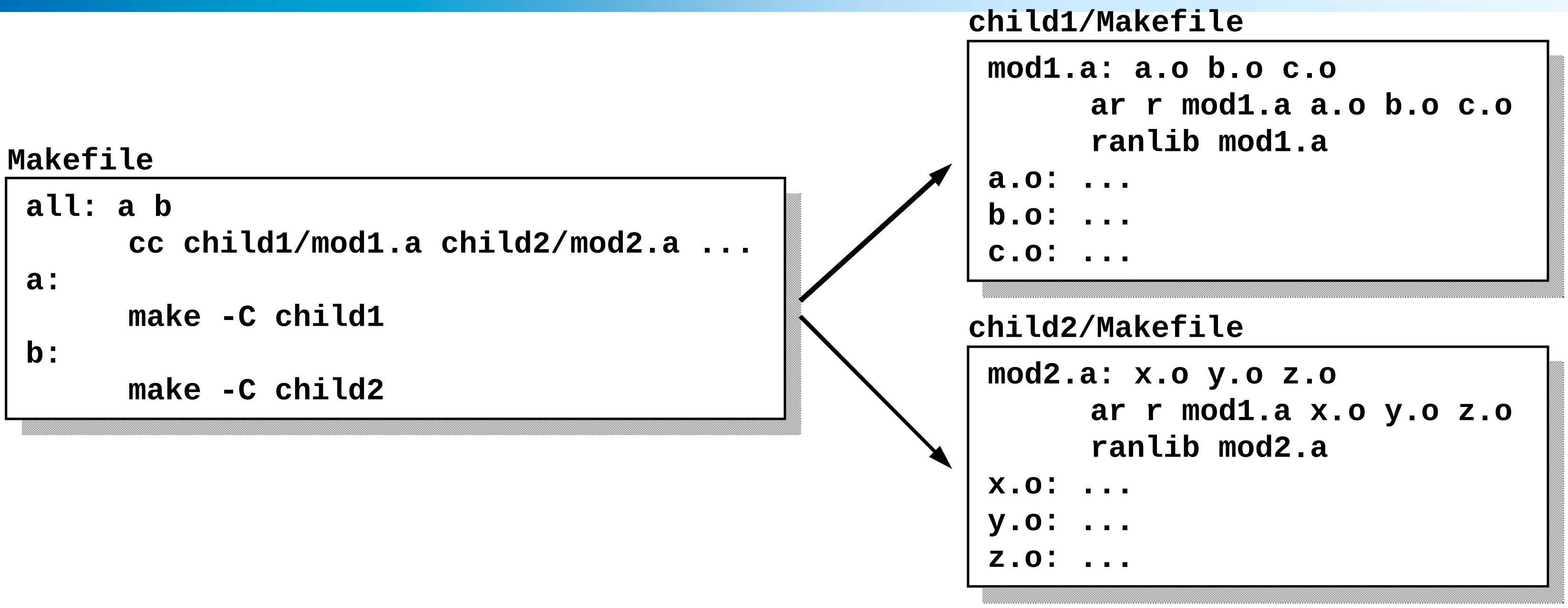
- **P2P file transfers offload 20-25% of outbound traffic:**
  - Take advantage of inexpensive bandwidth within switch
- **Just-in-time compression cuts traffic 2.5-3x:**
  - Match network bandwidth to disk

File System Optimization

electric cloud

- **Highly parallel builds stress build machine's file system :**
  - Average bandwidth as high as 10-20 MB/s
  - ClearCase? High latency
- **All disk I/O passes through Electric Make: opportunity to manage read & write concurrency**
  - Single disk? Concurrency causes extra head motion
  - Network file system? More concurrency hides network latency
- **Metadata caching improves ClearCase performance significantly**

Recursive Makes



Makefile

```
all: a b
      cc child1/mod1.a child2/mod2.a ...
a:
      make -C child1
b:
      make -C child2
```

child1/Makefile

```
mod1.a: a.o b.o c.o
      ar r mod1.a a.o b.o c.o
      ranlib mod1.a
a.o: ...
b.o: ...
c.o: ...
```

child2/Makefile

```
mod2.a: x.o y.o z.o
      ar r mod1.a x.o y.o z.o
      ranlib mod2.a
x.o: ...
y.o: ...
z.o: ...
```

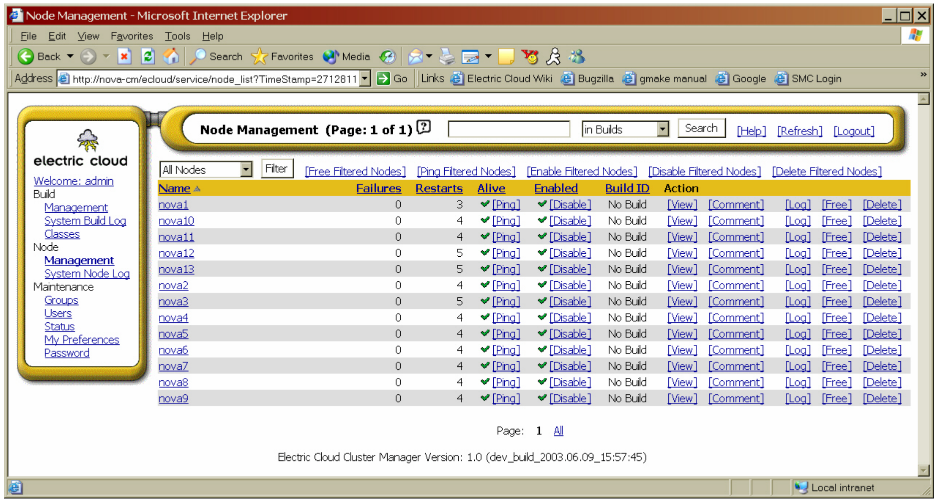
- **Gmake: separate gmake invocation for each Makefile:**
  - Hard to extract & manage concurrency
  - Can’t manage dependencies across Makefile
- **Electric Make: merge Makefiles**
  - Recursive makes return immediately with parameter info
  - Top-level emake manages multiple *make instances*

Compatibility

- **Plug-compatible with GNU Make, Microsoft NMAKE, Sun make**
  - Change ‘gmake’ or ‘nmake’ to ‘emake’ in build scripts
  - Identical command-line options
  - Identical results (except builds run faster)
  - Identical log file output
  - Typically a few Makefile changes to maximize speedup

Manageability

- **Web-based administration**
  - As easy to manage many nodes as 1 node
- **Can be used by entire team:**
  - Supports multiple simultaneous builds
  - Priority system for node allocation
- **Robust: automatic fail-over on node failures**



http://nova-cm/ecloud/service/node_list?TimeStamp=2712811

Node Management (Page: 1 of 1)

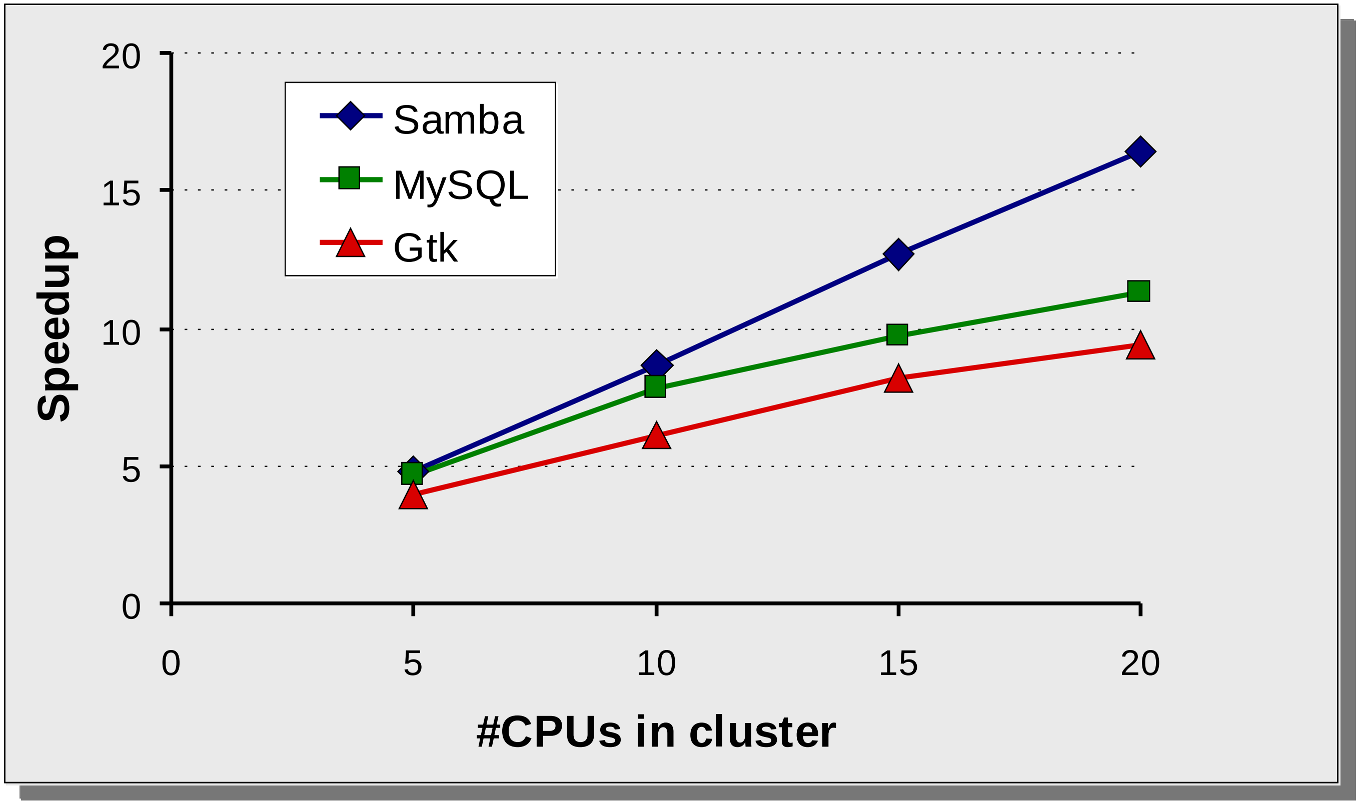
Name	Failures	Restarts	Alive	Enabled	Build ID	Action				
nova1	0	3	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova10	0	4	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova11	0	4	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova12	0	5	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova13	0	5	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova2	0	4	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova3	0	5	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova4	0	4	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova5	0	4	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova6	0	4	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova7	0	4	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova8	0	4	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]
nova9	0	4	✓ [Ping]	✓ [Disable]	No Build	[View]	[Comment]	[Log]	[Free]	[Delete]

Page: 1 All

Electric Cloud Cluster Manager Version: 1.0 (dev_build_2003.06.09_15:57:45)

Results: Open Source

electric cloud



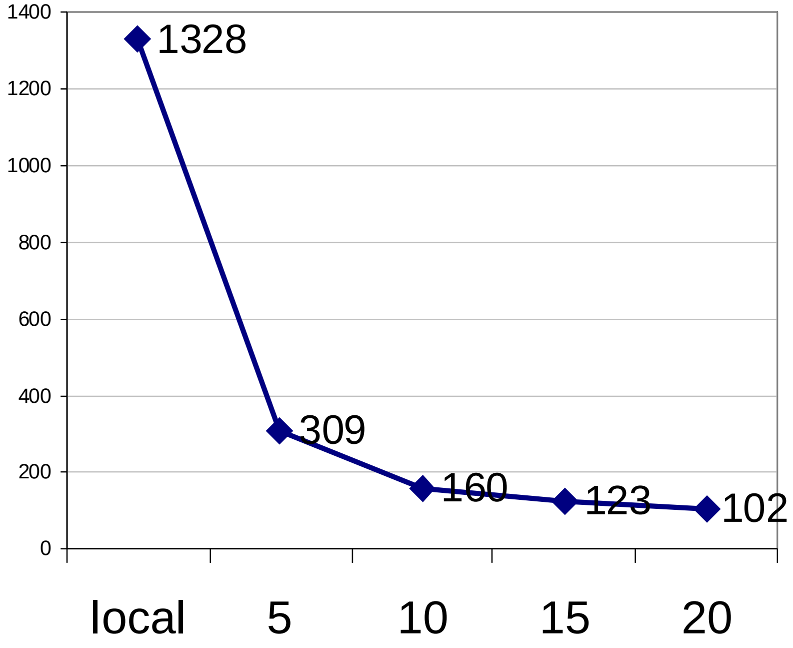
	Local	20 CPUs	Speedup
Samba	952s	58s	16.4x
MySQL	1400s	124s	11.3x
Gtk	891s	95s	9.4x

Results: Linux Kernel

- **Linux Kernel 2.6.1**
- **Make bzimage + modules**
- **2.8 GHz Xeon, 1 GB RAM, IDE Drive**

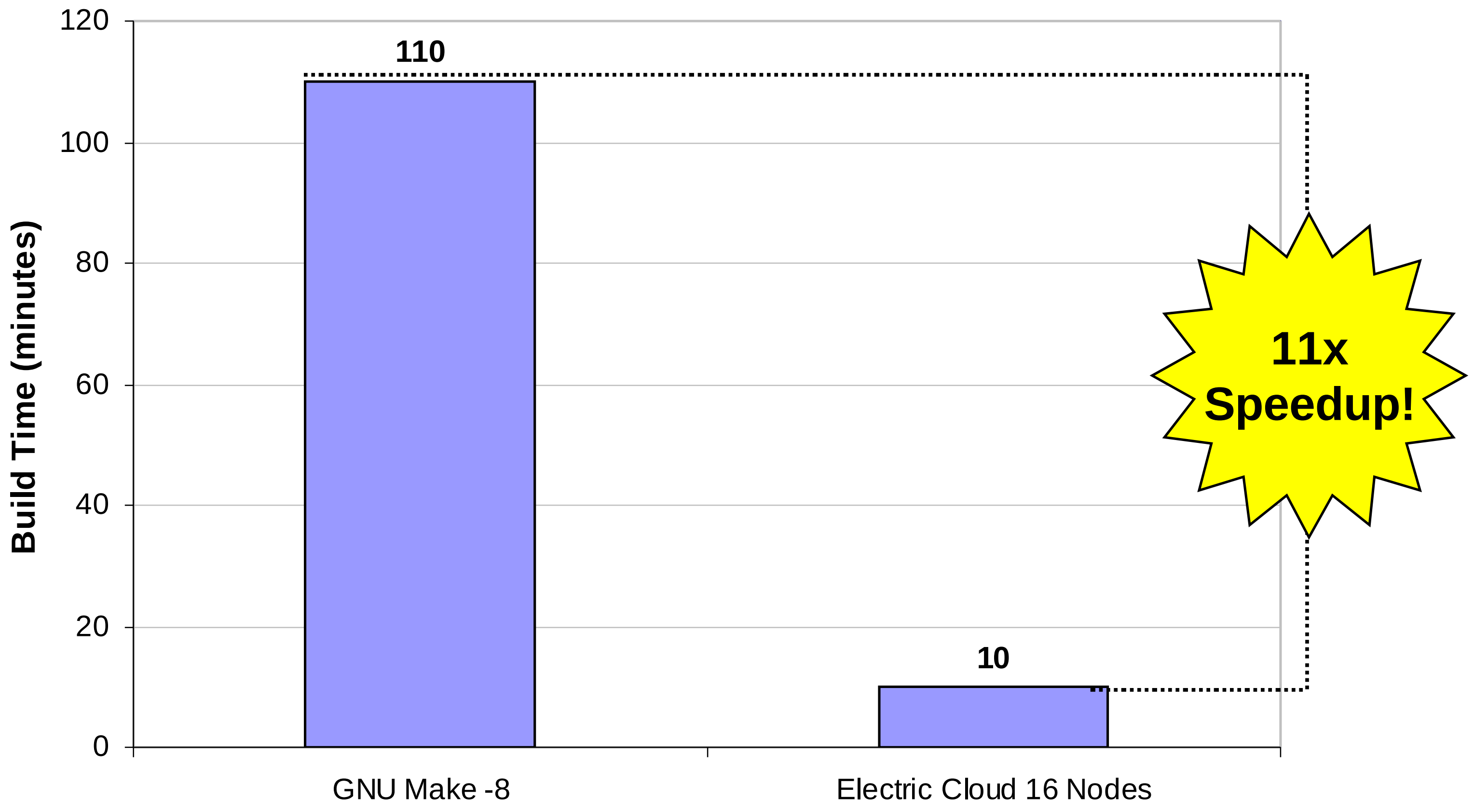
	Build Time [mm:ss]	Speedup
Local	**22:08**	
5 nodes	**5:09**	**4.3x**
10 nodes	**2:40**	**8.3x**
15 nodes*	**2:03**	**10.8x**
20 nodes*	**1:42**	**13.0x**

* Projected build time



Telecom Equip. Vendor

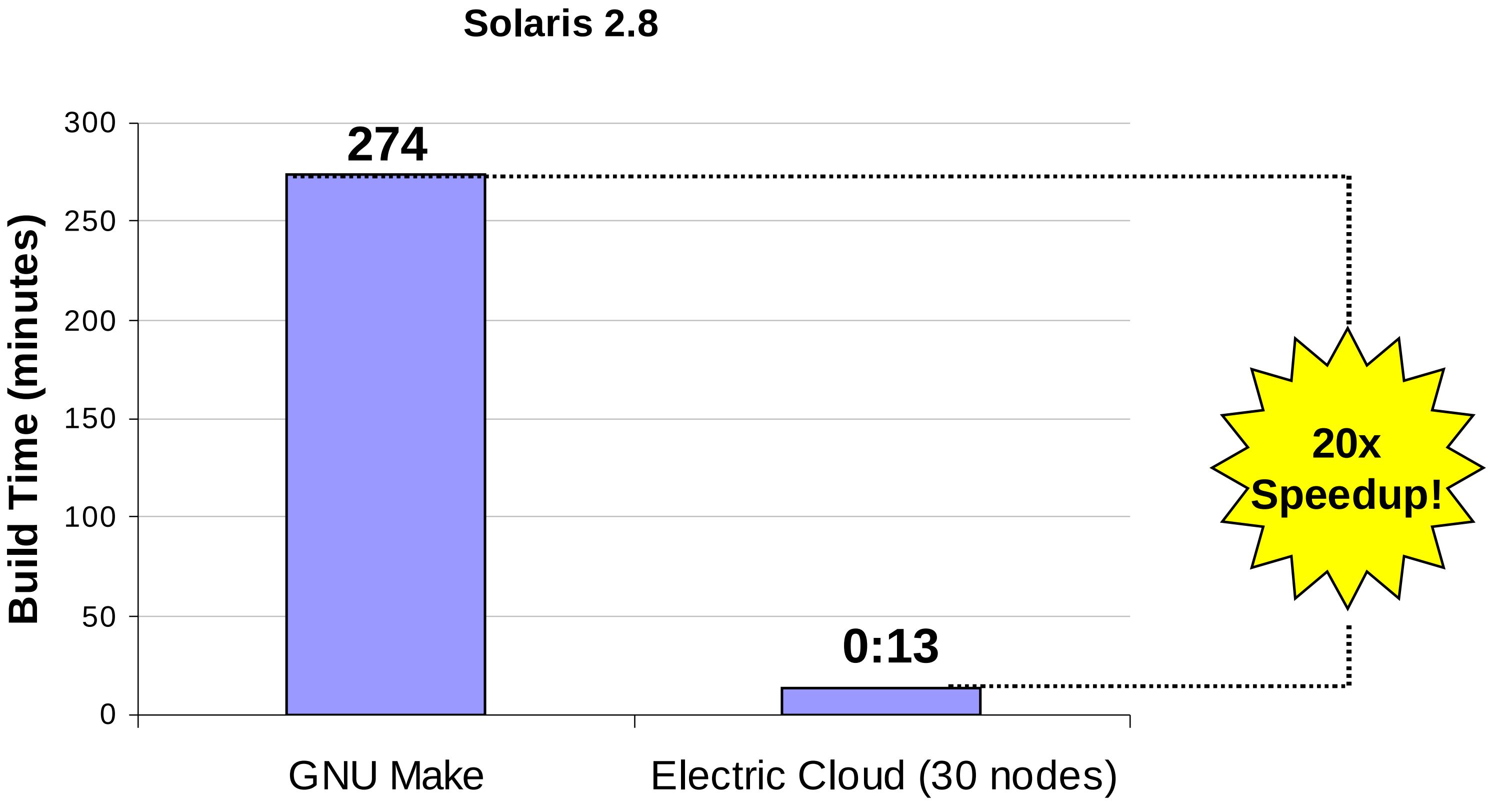
electric cloud



Impact: 3 week savings out of an 8 month release cycle expected

Enterprise Software Co.

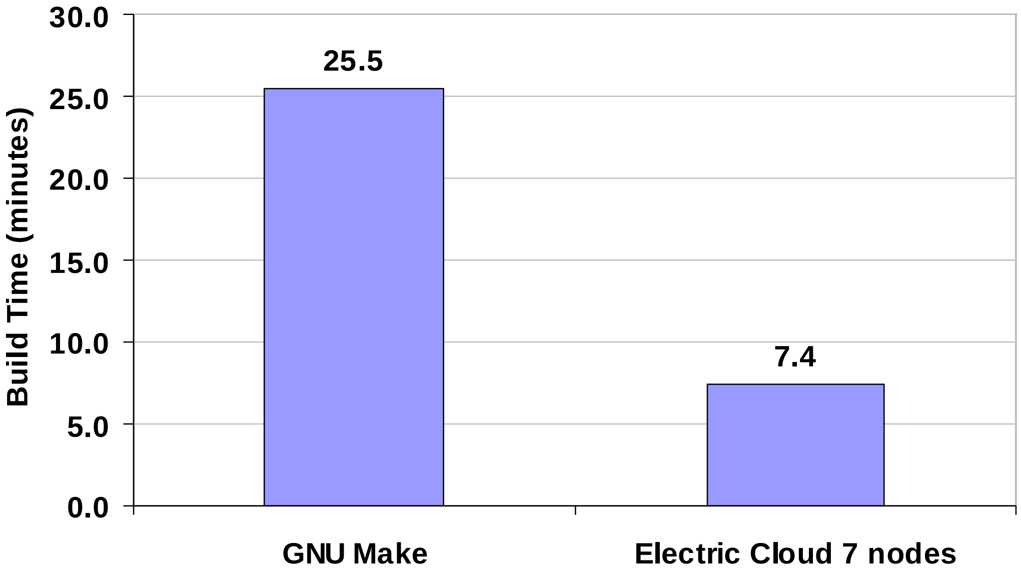
electric cloud



Impact: Enabled worldwide follow-the-sun development

Electric Cloud

electric cloud



- **We eat our own dog food**
- **Continuous build system:**
  - Start build and test cycle whenever changes are committed to the main branch

What about distcc?

- **Works with gmake –j**
- **Distributes compile steps to nodes**
- **Preprocesses code on make machine:**
  - Preprocessed code is self-contained: eliminates virtualization issues

distcc vs. Electric Cloud

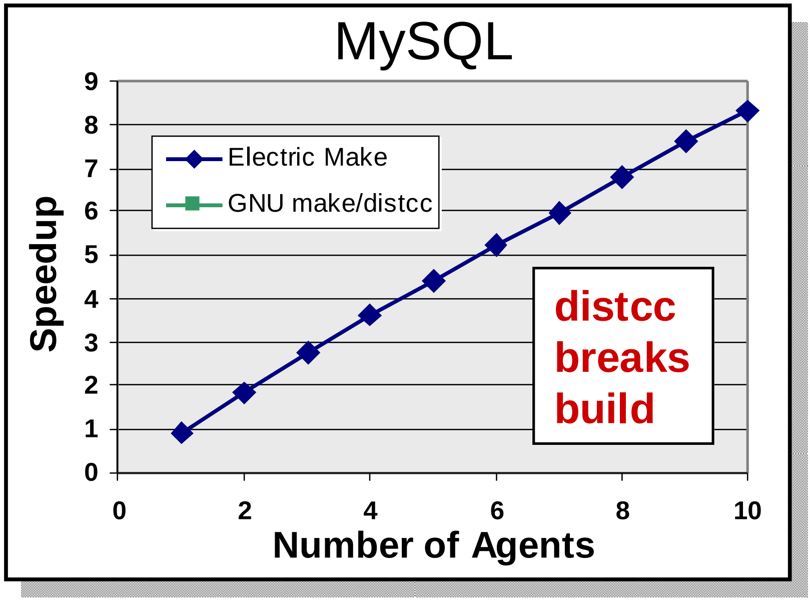
distcc:

- Free
- Works with other build tools (SCons?)
- Portable
- Compiler-specific (gcc)
- Less scalable:
  - Only distributes compiles; preprocessing centralized
  - Missing dependencies break build
- Build log scrambled
- No cluster sharing facilities?

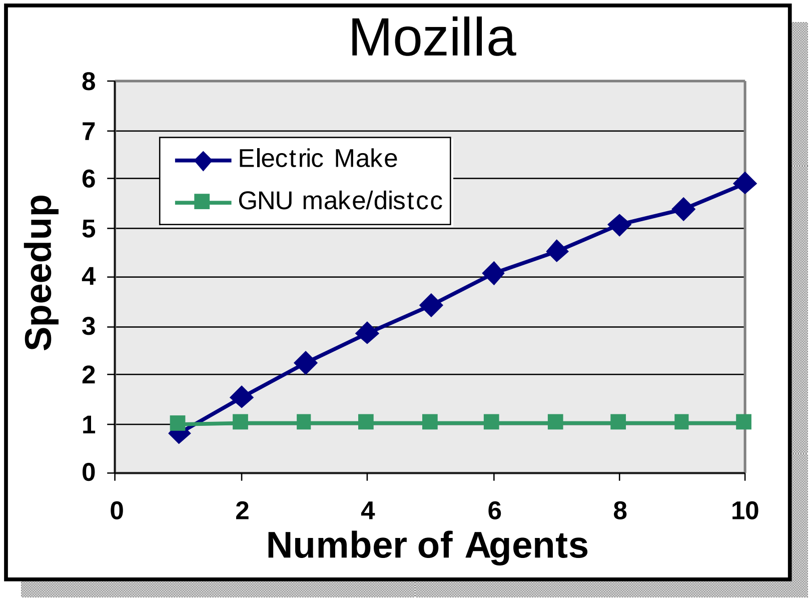
Electric Cloud:

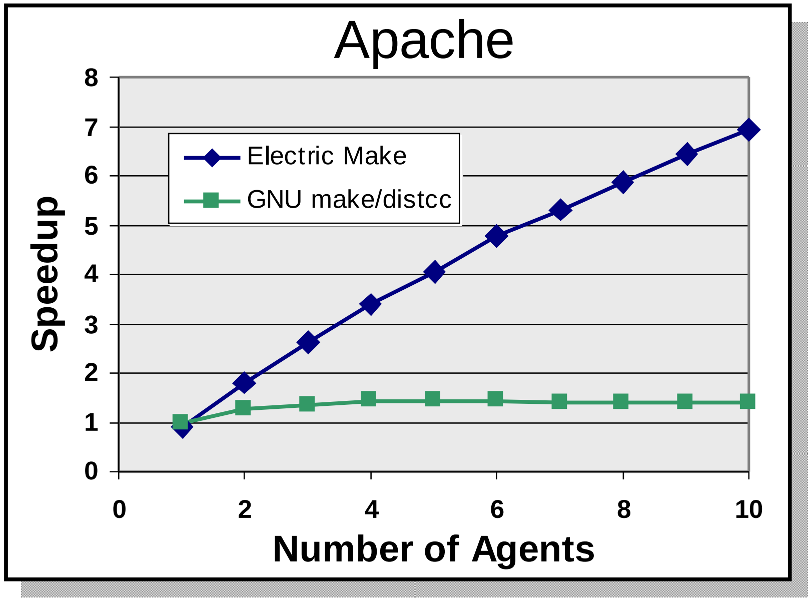
- Not free
- Only works with Make
- Windows, Linux, Solaris
- Works with all compilers
- More scalable:
  - Distributes all build steps (even Makefile parsing)
  - Deduces dependencies to avoid build breakage
  - Parallelizes sub-makes
- Build log in sequential order
- Cluster mgmt/sharing

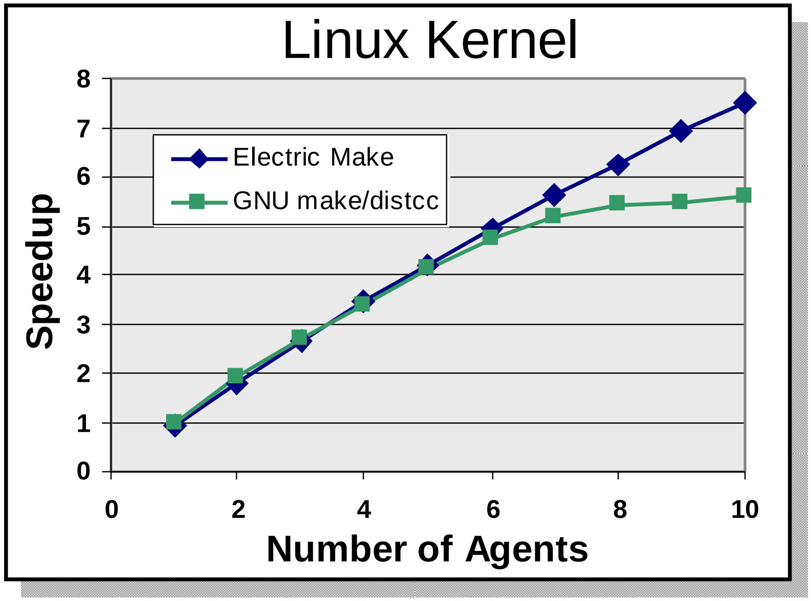
Electric Make vs. Distcc


MySQL

distcc breaks build


Mozilla


Apache


Linux Kernel

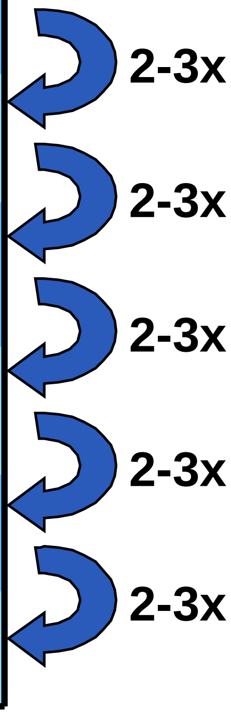
Performance Limits

electric cloud

- **File system on make machine**
  - ClearCase dynamic views particularly slow
- **Serializations within builds**
  - Linking slow on Linux
- **Make machine CPU not an issue**
  - Typically running at 30% utilization

Impact of 10-20x Speedup

Build Time	Impact
14 hours	Build doesn’t finish overnight
6 hours	Overnight build
2 hours	Multiple revs in a single day
30 min.	Full rebuild before checkin
5 min.	Little need to switch context
1 min.	No need to switch context



Electric Cloud can drop you two bands

Conclusion

- **No need to tolerate slow builds anymore**
- **Faster builds mean**
  - Faster time to market
  - Higher quality
  - Ability to do more with less

More Information

electric cloud

- **For more information or to answer additional questions:**
  - Visit our website: www.electric-cloud.com
  - E-mail: info@electric-cloud.com
  - Phone: 650-962-4777

electric cloud