The FreeBSD Package Cluster

Kris Kennaway
kris@FreeBSD.org

BSDCan 2005, Ottawa, May 13

Outline

- Goals of the package cluster
- Challenges
- Overview of cluster architecture and implementation
- Anatomy of package build process
- Optimizations
- Future work
- Summary

Overview of Ports Collection

- FreeBSD Ports Collection provides build framework for compiling, installing and managing third-party software
  - 12852 ports at time of writing
- Binary (precompiled) packages may be produced for easier installation on other machines
- 170 ports committers working on maintaining ports collection and managing submissions from user community

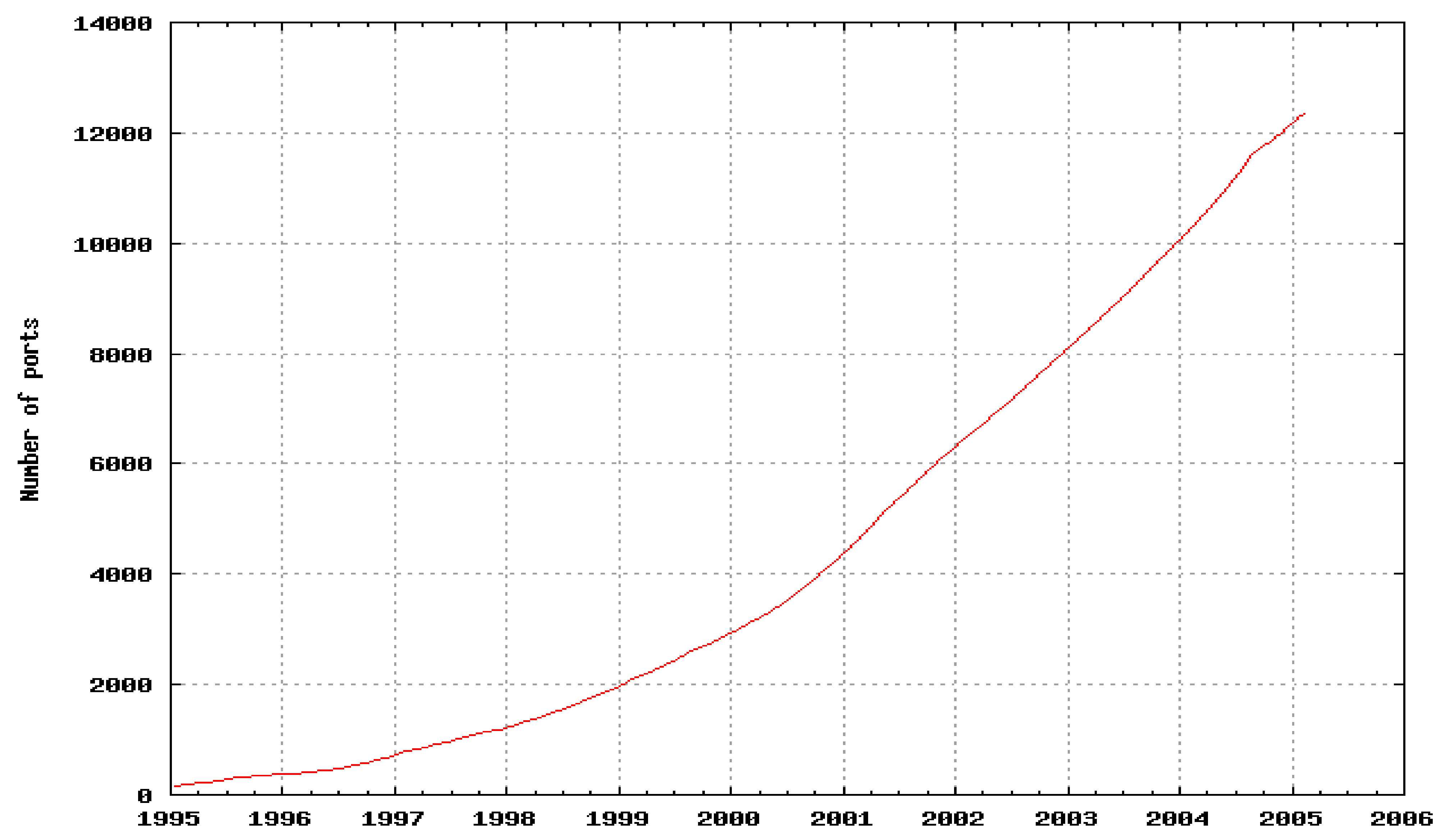
Goals of the package cluster

- Provide up-to-date packages for FTP and release distribution.
- Automated QA of FreeBSD ports collection
  - Test port/package compilation
  - Identify common errors
  - Semi-automated reporting to responsible parties
- Test bed for architectural development and large-scale changes to ports collection
  - Maintaining stability of ports collection for end-users is paramount
  - Ports collection contains all manner of weirdisms
- QA of FreeBSD development and stable branches
  - Exercises wide feature set and operational conditions; very good testbed for identifying bugs and focusing developer attention on problems.

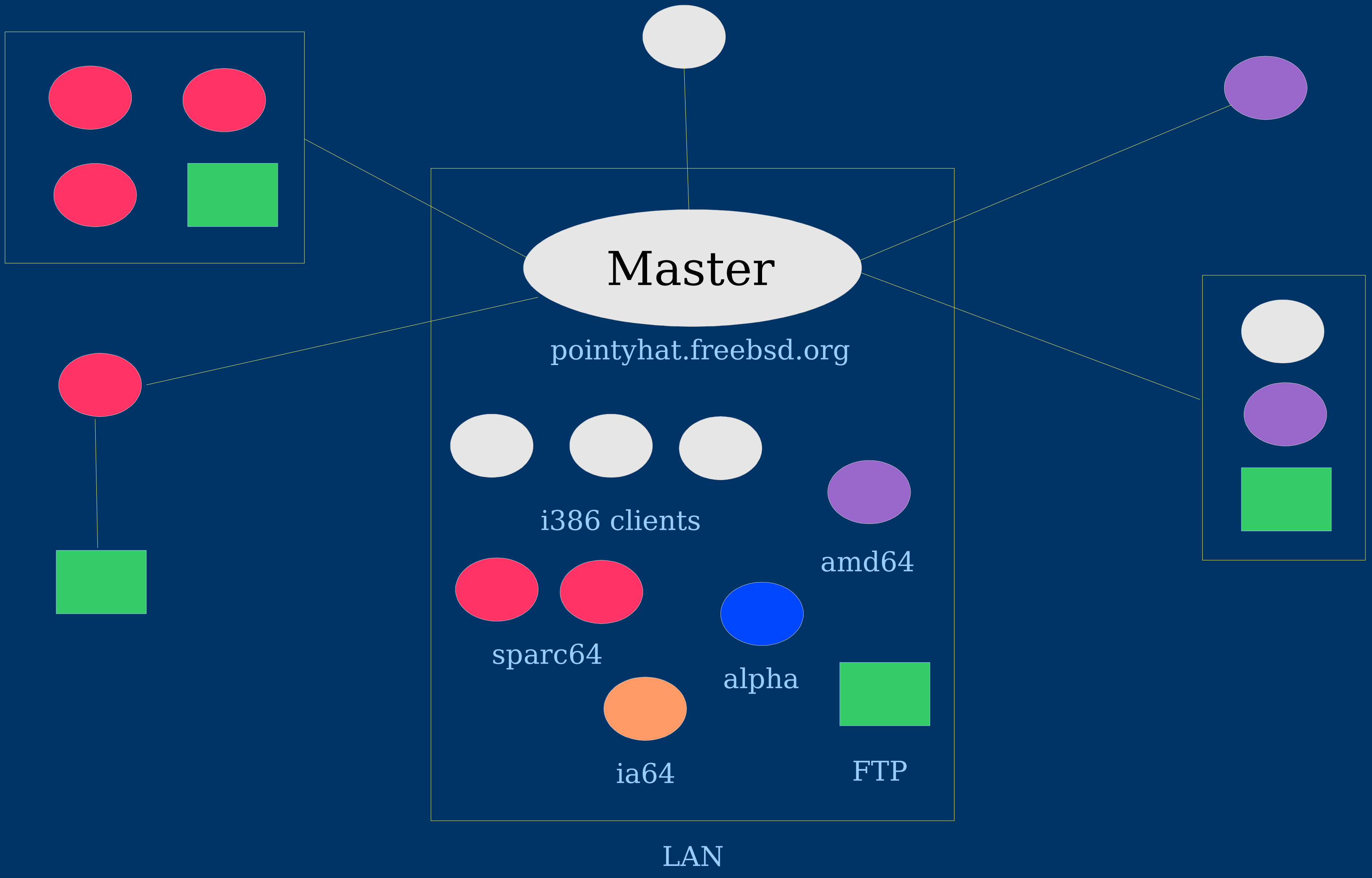
Challenges

- Large number of ports (>12852)
- 3 supported branches (4.x-STABLE, 5.x-STABLE, 6.0-CURRENT)
- 5 supported architectures (i386, alpha, sparc64, ia64, amd64)
- Rapidly changing ports collection (dozens of commits/day)
  - Large fraction of ports collection affected over short timescales
- Balancing package build and ports development uses of the cluster
- Rapid growth of ports collection

Growth of the ports collection



Schematic of cluster architecture



Build resources

- Master (pointyhat.freebsd.org)
  - Dual i386 p3 1.3GHz
  - 2GB RAM, ~280GB disk
- Clients
  - i386: 27 p3 800MHz, 512/1024MB
  - SPARC64: 12 clients (freebsd.org, .jp, .ca, .us)
    - 9 Ultra 10, 2 4-CPU E450, 2-CPU E420R, 12-cpu E4500
  - AMD64 (freebsd.org, .us)
    - 1.6 Ghz,512MB; 2GHz*4, 16GB; 2GHz*2, 8GB
  - IA64: 2 900MHz McKinley (freebsd.org)
  - Alpha: 5 DS10 (freebsd.org)
- Secondary test cluster (yahoo.kr); 2 i386 p4 2GHz

Cluster architecture and history

- Cluster built on shell scripts, standard UNIX tools (make, ssh, netcat,...) and some custom C code
  - Current implementation scales well enough with current machine resources
- Evolved continuously from original implementation by Satoshi Asami (ca 1999)
  - Significant changes and improvements over the past few years
- Evolutionary pressures from scaling of ports collection and cluster requirements
- Need to keep cluster in near-continuous operation limits windows for major redevelopment
  - Secondary test cluster (Y! Korea) useful for developing changes

Overview of the build process

- Build master prepares the build and initializes the client machines
- Jobs dispatched in parallel to available client machines
- Packages are built in separate chroots on the client
- Results of build are copied back from slave to master
- Master produces reports (webpage, email) of package build status
- Packages post-processed and published
- Multiple simultaneous package builds
  - Different architectures, branches
  - maximize resource utilization

Configuration of build server

- Job ordering uses Makefile constructed from package dependency data
  - Ensures correct ordering of dependencies
  - Automatically handles package build failure
- Communication with clients over ssh
  - Suitable for local/remote clients
  - All communication initiated by server
- HTTP server for client fetching of packages
- NFS server for local client machines (netboot)
- Scheduler tracks job load on client machines
  - Detect offline machines
  - Package builds preferentially distributed according to machine capability and load
    - Avoid over/underloading machines

Configuration of the build clients

- Netbooting where possible for ease of maintenance
- Typically run FreeBSD-CURRENT or -STABLE
  - Require certain minimum feature set
  - QA of FreeBSD active branches
- Build chroots populated with image of target FreeBSD world (4.x/5.x/6.x)
  - *Deliberately mismatched* kernel/world in chroot
    - Allows simultaneous builds for different FreeBSD branches on same machine
    - No need to reboot client and maintain separate installations
    - Some kernel-sensitive binaries copied in from host environment
    - FreeBSD backwards compatibility takes care of the rest

Preparing a package build

- Update ports/src/doc trees
- Build an INDEX file
  - Records package name/port directory mappings
  - List of package dependencies
- Build a list of known-unbuildable ports
  - Ports marked IGNORE/FORBIDDEN/... will never be built because of known limitations (e.g. unsupported version; security vulnerability; ...)
  - Ports marked BROKEN are built infrequently to test whether breakage still exists
- Prepare directories on master (log files, packages)
- Construct makefile from INDEX dependency list
  - used to order job dispatches
  - ~13MB, 38000 targets

Incremental package builds

- To avoid unnecessary rebuilding, most package builds are *incremental*
  - Compare old and new INDEX files
  - Identify packages with changed version string, or changed list of dependencies
  - Remove these packages from the previous package set
  - Only these packages, and those depending on them, will be rebuilt automatically by the master Makefile.
- Incremental builds often only take a few hours
- Full rebuilds less often to catch unfetchable ports and those broken by FreeBSD base system changes

Preparing the client machine

- Remove stale build chroots
- Refresh client copy of cached data
  - Tarball for populating chroots
  - Remote clients: copy of ports/src/doc tree and build scripts are refreshed with rsync
- Ensure that all resources are available
  - squid, disk space
- Ready to begin dispatching package builds!

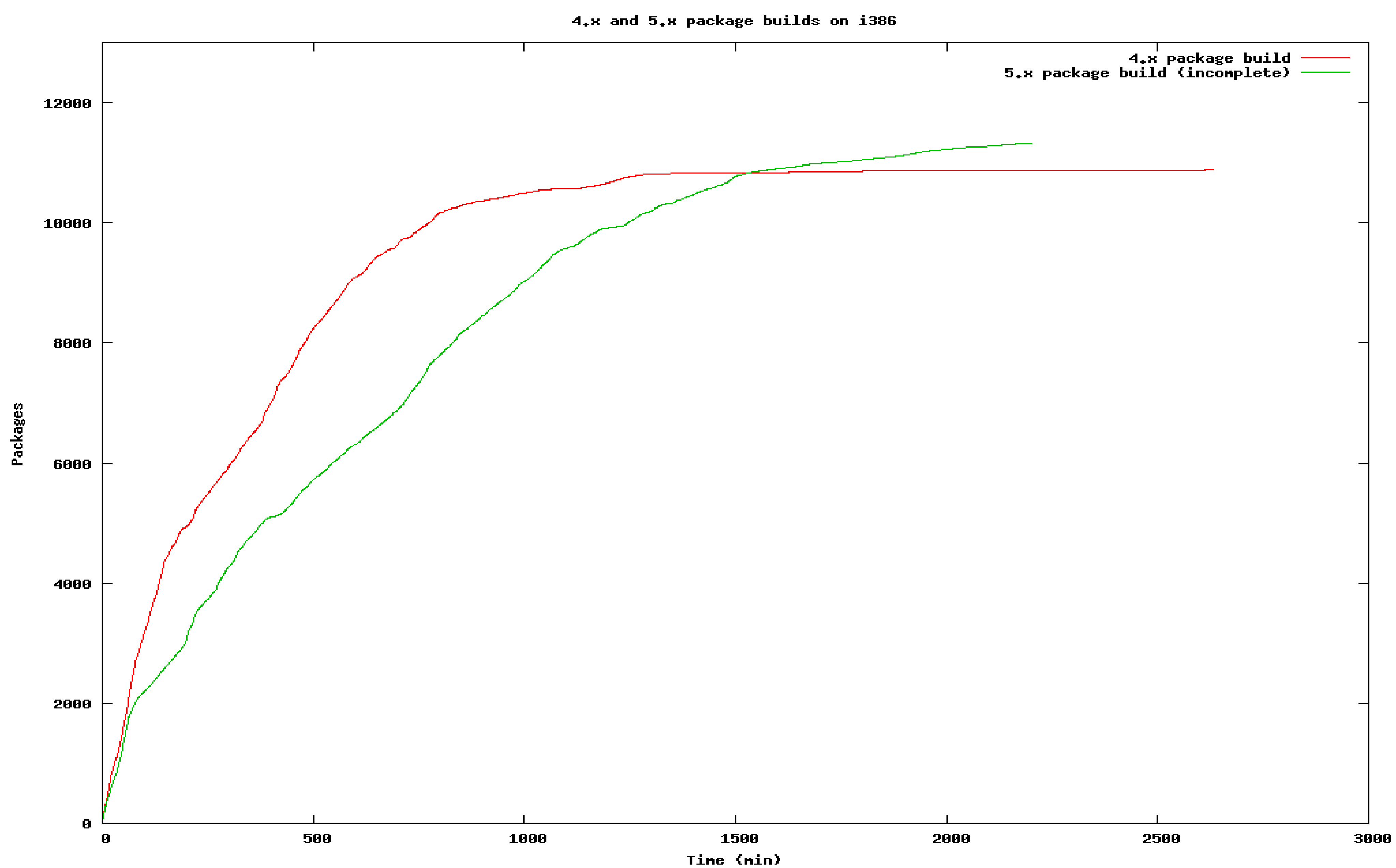
Anatomy of a package build (I)

- Build machine with free job slot is selected
  - Build concurrency >= # CPUs for optimal resource usage
- Free chroot is claimed for use, or a new chroot is created and populated
- Job dispatched to client over ssh
- Ports/src/doc trees are mounted inside the chroot
  - NFS from a common local server
  - NullFS from a local filesystem image
- For each build stage (fetch, extract, patch, build/install), package dependencies are fetched via HTTP from master
  - Squid cache used to reduce network traffic
    - Many packages reused (e.g. Perl, X libraries), so up to 90% hit rate

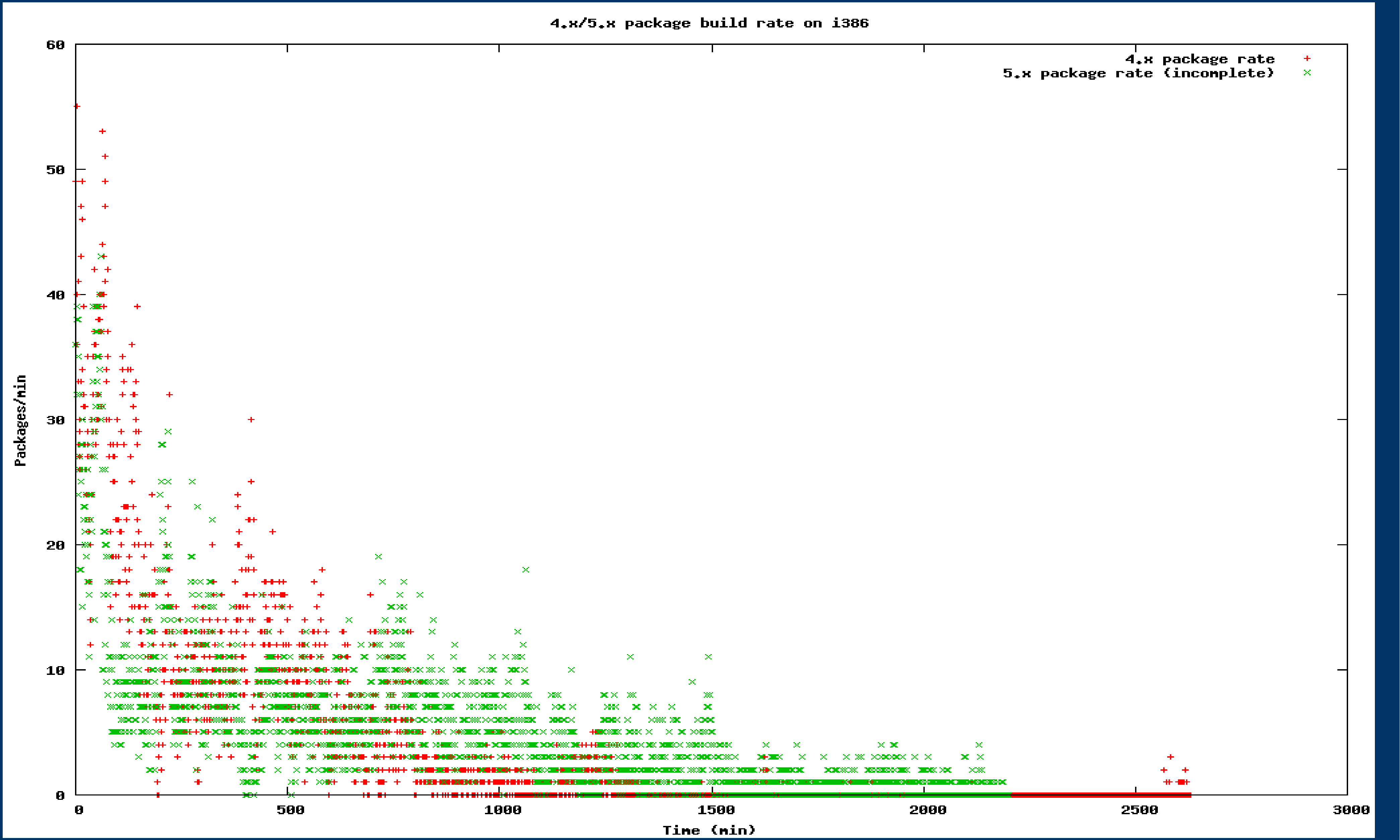
Anatomy of a package build (II)

- Package dependencies added
- Build stage is executed (fetch/extract/patch/build)
- Package dependencies are *removed*
  - Verifies that the dependency list is correct at each stage
- If build completes, package is created
- Packing list is verified
  - All files listed in packing list were installed
  - No installed files that are not listed in packing list
- Build chroot is cleaned
- Build master copies back results of build (success/failure logs, package)
- Chroot on client is released for new build

Package build progression



Package build rate



Build post-processing

- Non-redistributable packages are removed if package set is to be uploaded to FTP/distributed on CDROM.
- INDEX post-processed to remove unbuilt packages
  - e.g. for use by sysinstall
- Checksum of packages constructed
- Packages rsync'ed to FTP site if requested
  - incremental builds: only new/changed packages
    - Cuts down on FTP mirror load
- Port distfiles rsync'ed to FTP site if collected

Package build summary data

- http://pointyhat.freebsd.org
  - tracks results of package build as it progresses
  - maintains history of broken ports with logs, for each supported FreeBSD version and architecture
    - classified by error type
  - logs of successful builds
  - useful for port maintainers, committers and users
- Feeds other databases (http://portsmon.firepipe.net fenner's distfile survey, ...) providing other views of this dataset
- Email reports of individual port build failures are post-processed by Mk I Eyeball to weed out false positives, and forwarded to responsible party for action

Optimizations (I)

- Kernel optimization
  - 6.0 much better than 5.x, particularly on 5.x (mpsafevfs)
    - Still in development though (i.e. some bugs on SMP)
- Cache, cache, cache!
- NullFS > NFS on busy networks/servers
    - Time trade-off for initial rsync
- vfs.nfs.access_cache_timeout=300
  - NFS data is static throughout the life of the build
- Squid proxy
- Local FTP distfile mirror where possible
- Maintain constant build load (don't over/underload machines)

Optimizations (II)

- Memory disk (md) instead of disk-backed FS for package builds
  - Dramatically cuts disk writes, even for swap-backed md
  - Build each port in separate md on SMP
    - Better concurrency from multiple md kernel threads, especially with mpsafevfs on 6.0
    - Able to completely saturate 12-processor E4500 on 6.0 (i.e. very little Giant contention)

Future work

- Repeated pkg_add/pkg_delete during build stages is time-consuming
  - Better: leave package installed in a chroot, and *relocate the build directory between chroots* instead of adding/removing the package in the same chroot
  - Trade time (is money) for disk (is cheap)
- Explore use of ccache for caching compilation
  - Works well for single machines, but need to deal with build locality
- Better management of transient build resources
  - Deal with machines coming/going
  - Network outages
  - Machine reboots

Summary

- High-performance, custom purpose distributed cluster for building binary packages from FreeBSD ports collection
- All components freely available
  - /usr/ports/Tools/portbuild/
- Documentation available
  - http://www.freebsd.org/doc/en/articles/portbuild/index.html
- More machine resources always welcome
  - Preferably several fast machines hosted by well-known company/community member
- A major source of QA for FreeBSD Ports Collection and the FreeBSD Operating System.
  - stress-tests FreeBSD under real-world loads