tl;dr

Let’s have pbulk support cloud tools and dynamic worker up-/downscaling.

Disclaimer: This is a proposal.

I have not actually written any code.
Introduction: Bulk Builds in pkgsrc
This is NetBSD after all, so there are three bulk build tools:

1. „the old bulk build tools“ (now gone)
2. distbb (few, if any, users)
3. pbulk
   This is what we’ll focus on
Distributed pbulk

- Master + n workers

- There are two pkg trees involved: the „outer“ one (/usr/pkg_bulk, w/ pbulk package) and the build one (as a bootstrap kit)

- All workers share the same pkgsrc tree (e.g. via NFS)

- There are two phases, scan and build
How to pbulk

1. Bootstrap inner tree, create bootstrap kit (tar.gz)

2. Bootstrap outer tree, install pbulk, edit config file (add IP addresses of all workers), export tree to workers
   - Be very careful about the paths in that config!

3. Make sure that the master can ssh to workers as root, with no password (yeah right)

4. Run /usr/pkg_bulk/bin/bulkbuild, wait …
What happens now

• Master starts a TCP listener and runs ssh to each worker

• Scan phase: get dependency tree etc, max parallelism

• Build phase: master gives a package to build
  • worker deletes entire pkg tree, unpacks bootstrap, installs deps from packages, builds, tries install and uninstall
• Number of workers is fixed.
  • even though you could run hundreds or just one at different times during the build

• Can run on public clouds, if you do all the work to set it up.

• (Also: TNF pkgsrc bulk builds are slow.)
Cloud support?
Rationale

• "If you cannot use binary packages, then you should build your own using lightweight VMs." – joerg@

• A public cloud is much more than a dumb VM provider!

• Cloud storage (e.g. S3) to replace NFS

• APIs to create/start/stop/destroy VMs on demand

• perhaps: hosted work queue (EBS)
Target Systems

- Amazon AWS (xen-based, excellent NetBSD support)
- Google Cloud (usable in NetBSD-8)
- Joyent Cloud (good for SmartOS)

- perhaps use an abstraction layer?
Revised procedure

• A script to create an AMI for a worker VM: boots directly into pbulk, master IP from metadata
  • worker uploads packages and logs to S3

• Master can be a tiny, cheap VM, beefy workers

• could use high max parallelism (e.g. 100, or ∞), master scales up/down as needed

• No worker VM is ever idle!
Changes and extensions

• Master must detect that it can shut down workers

• If master could detect that a build is stuck or worker went away: could use cheaper, preemptible resources

• Worker could create an index fragment

• Signing could be a separate service

• Could use a URL fetch service to fetch distfiles, upload distfile to storage after fetch
External work queue?

- Managed work queue service: App Engine Task Queue, hosted ZeroMQ, etc.
- Master stuffs names of pkgs to build into queue, workers fetch from there
- Stretch goal: serverless master (e.g. Lambda)?
Conclusion
Call For Action

• Let’s leverage the synergies of cloud computing ;)

• Please tell me why this is a Terrible Idea™ (I don’t think it is)

• If you think this is useful, consider helping to implement it