Please ask questions via the mobile app!

Engage

BERLIN INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

Follow us on Twitter @GOTOber

Towards Cloud-centric Development

What if we stopped treating the cloud like traditional servers?

Peter van Hardenberg
@pvh

“If I asked the people what they wanted…

…they would have told me **faster horses**.”

platform-as-a-service

a machine for turning code into useful applications

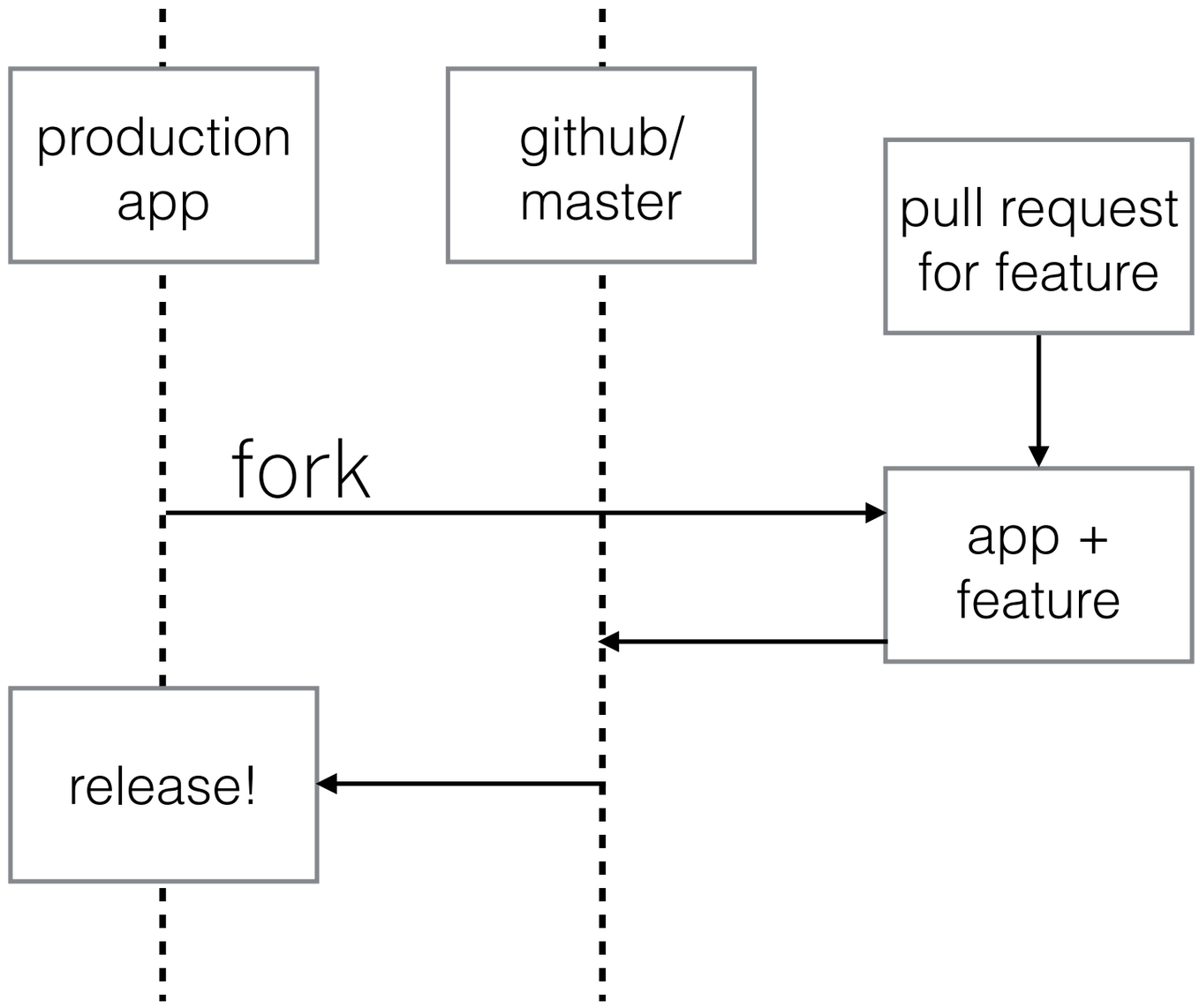

I DON'T ALWAYS TEST MY CODE
BUT WHEN I DO I DO IT IN PRODUCTION
memegenerator.net

Production on Demand

fourchette

"test your GitHub PRs against a fork of your Heroku app"

— RainforestQA 



What if you could automatically test in production?

Pre-abstracted

- Provision servers
- Design release process
- Develop monitoring capabilities
- Perform capacity management / scale projection
- Substantial per-app cost
- **Conclusion**: reduce cost by reusing architecture

Abstracted

- `git push heroku master`
- significant reduction in day-to-day operations
- **Conclusion**: dev & production costs are reduced

Post-abstracted

- Individual applications approach zero overhead
- Microservice architectures become feasible
- Clone whole running applications!
- **Conclusion**: decompose application architecture

Microservices love PaaS.

postgression

http://www.postgression.com/

stateless databases

for every test run... or whatever

Databases on Demand

Un-abstracted

- Predict required scale up-front
- Managed by in-house specialist
- Notoriously fragile
- Precious snowflake

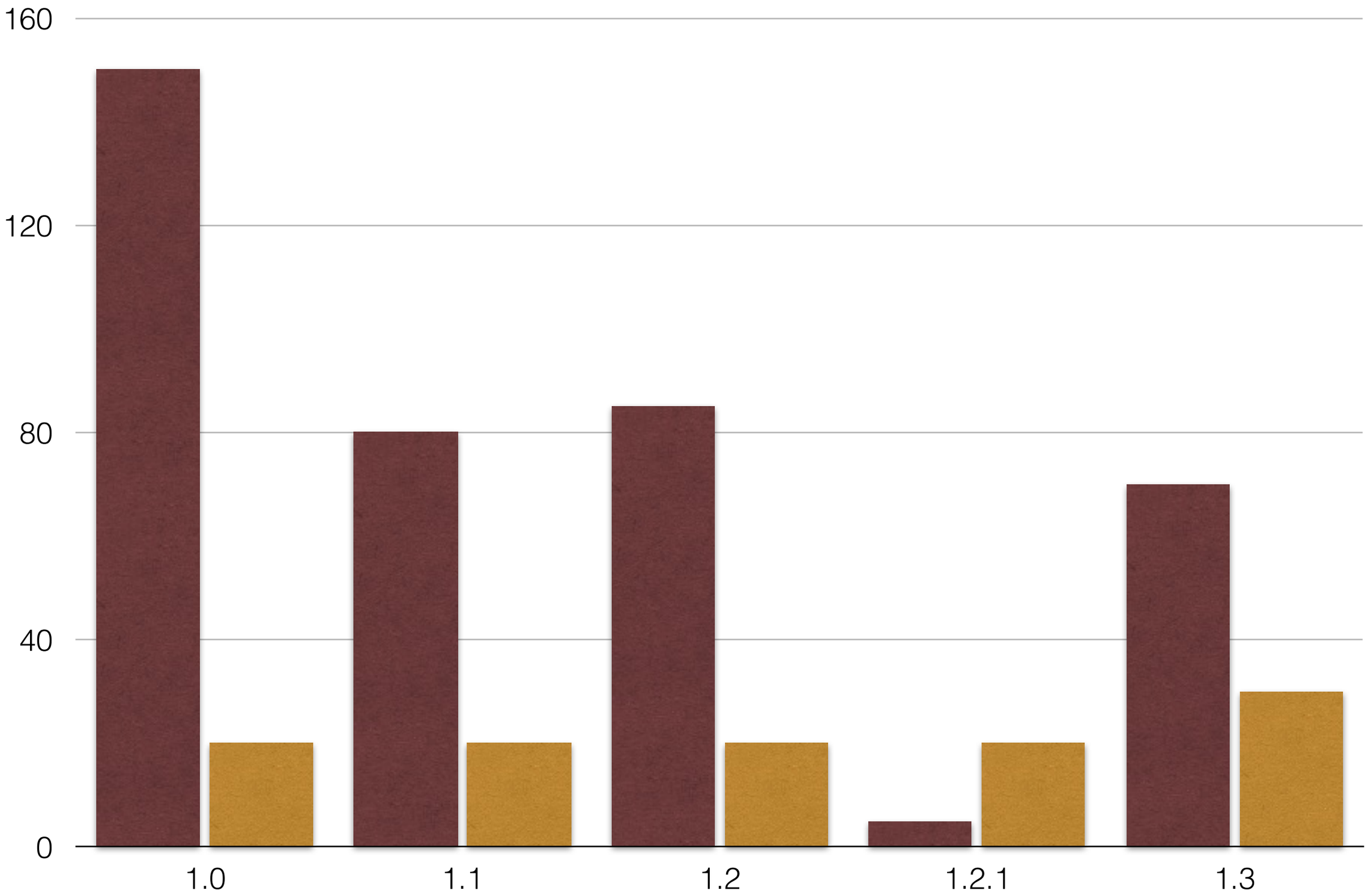
Abstracted

- Managed database on demand
- Take advantage of provider experience
- In-house data gurus focus on application
- Developers have parity with production

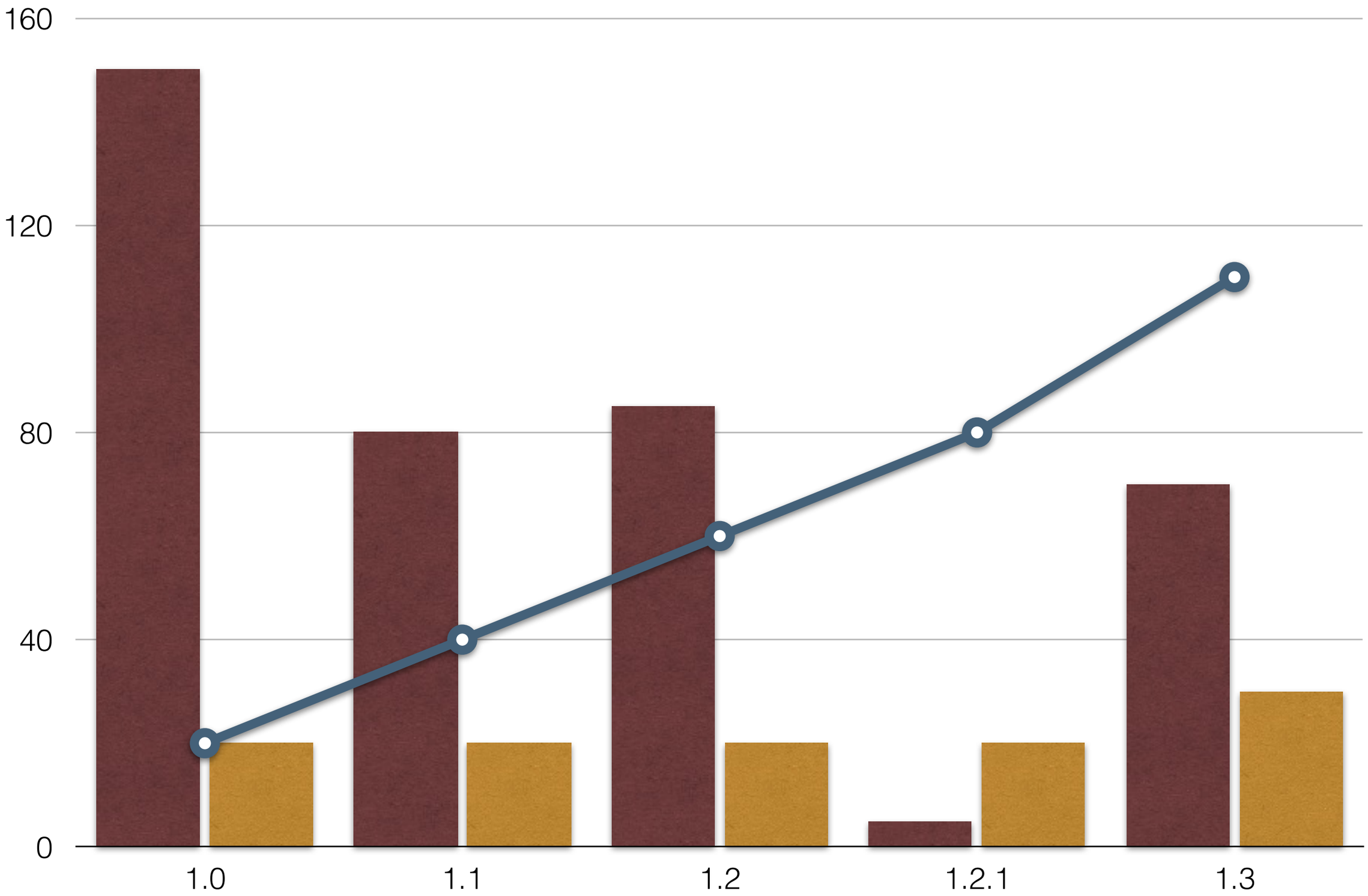
Post-abstracted

- Separate databases for separate concerns
- Respond to scale as it happens
- Take easy advantage of sophisticated infrastructure
- Clone production to test migrations
- Replicas for reporting
- Disposable databases for testing in parallel

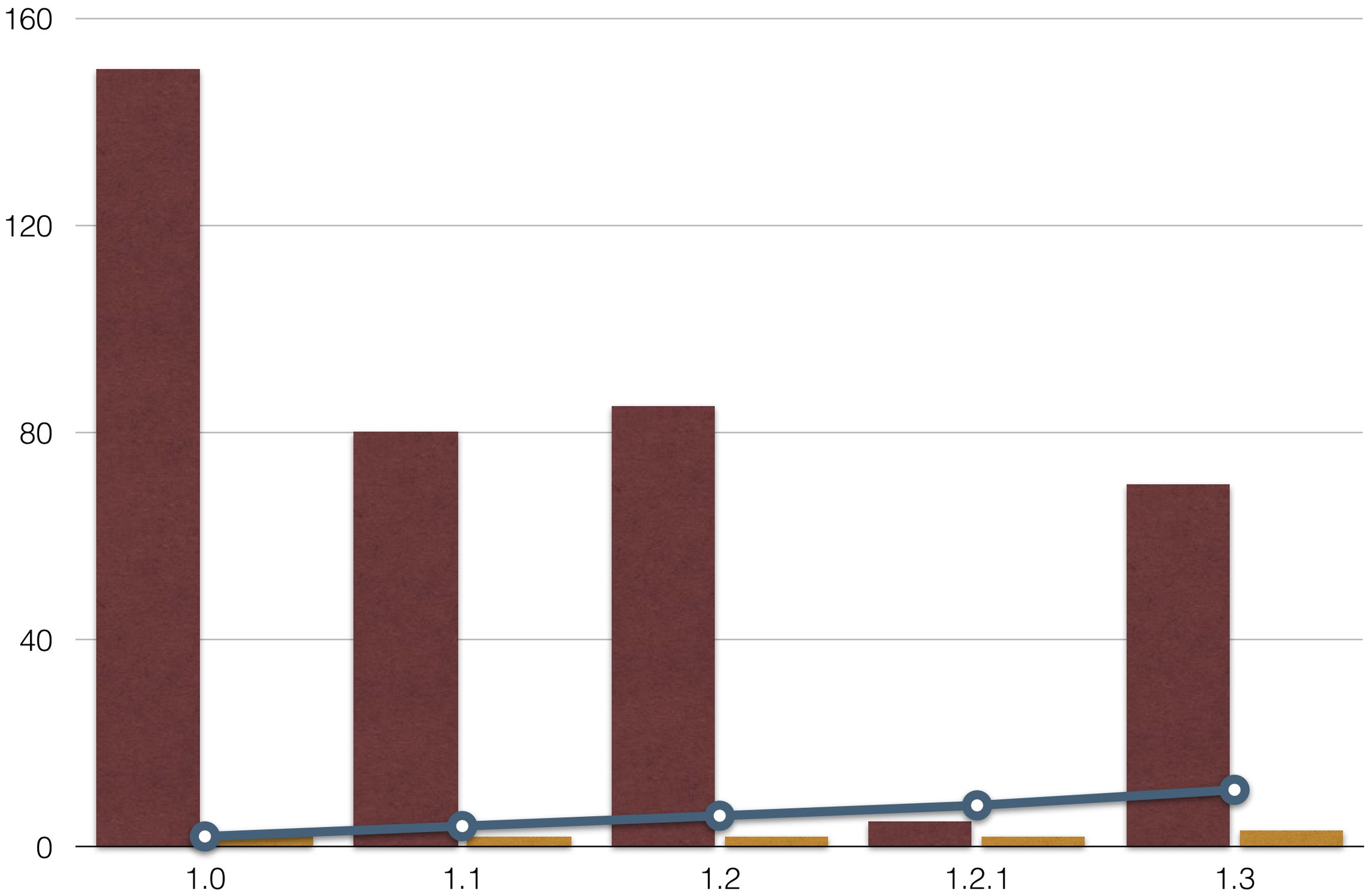
The hidden cost of releasing software



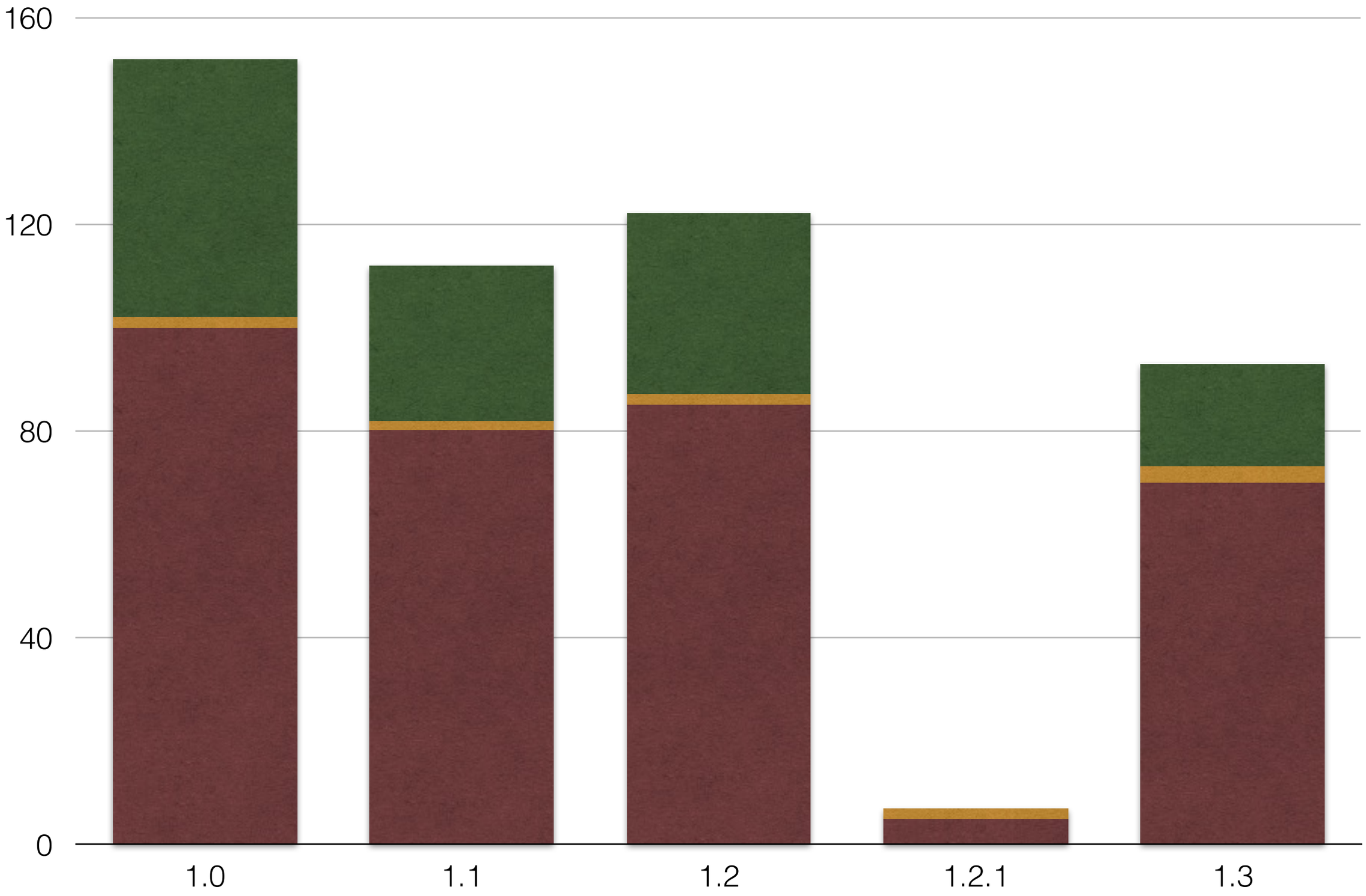
Big Releases



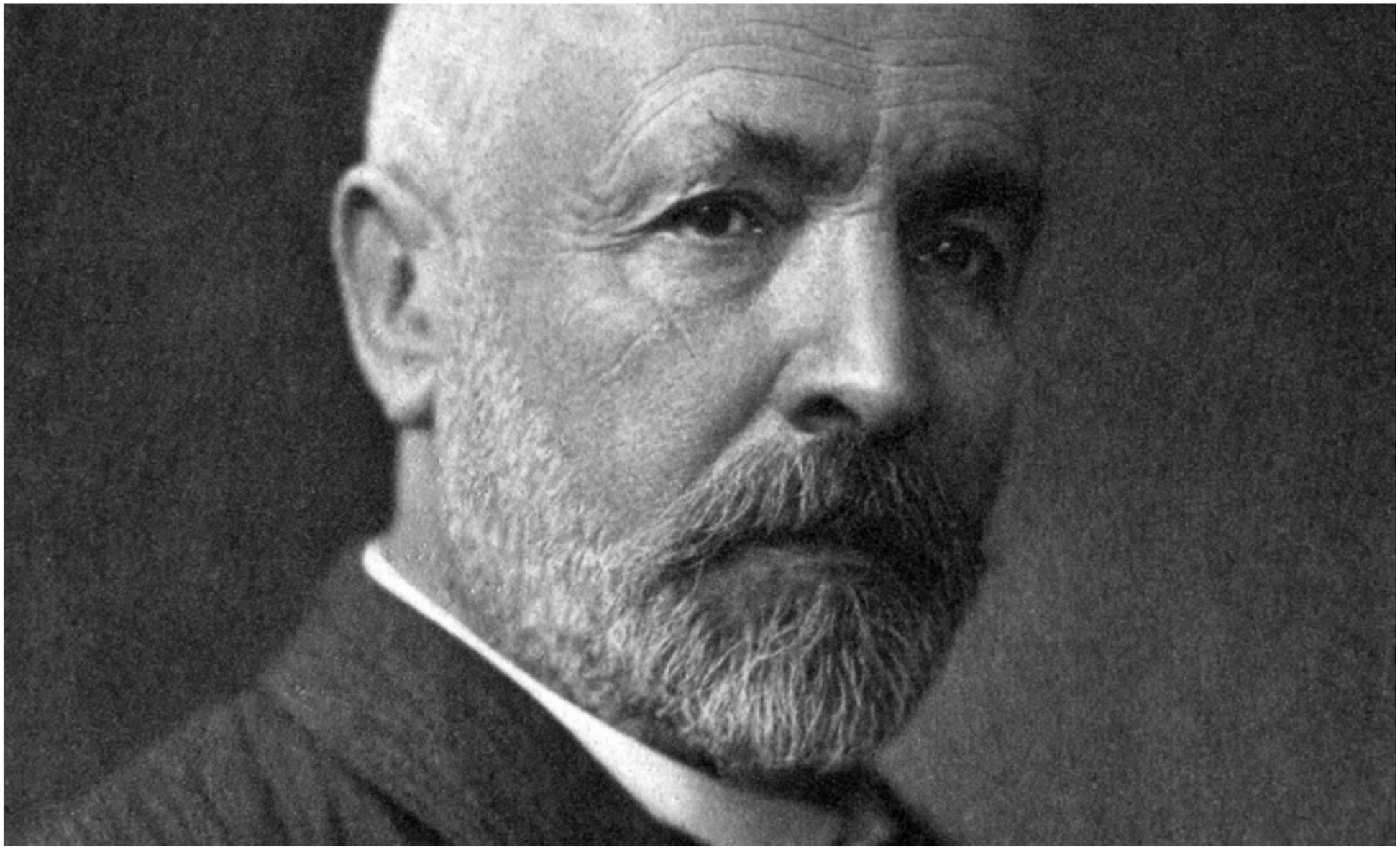
Big Releases



Big Releases



Hidden Integration Costs



let’s get mathematical

(I apologize to any actual mathematicians in the room.)

$$V = nF + n^2I + R$$

Cost of Releasing a Version

F: cost per feature
I: integration coefficient
R: cost to release

$$S = \sum_{t} (nF + n^2I + R)$$

Total Cost of Software

$$\lim_{R \to 0}(S)$$

$R > 0: n > 1$

$R = 0: n = 1$

reduced release overhead
makes small releases more efficient

Operational Overhead

- Certification processes
- Database migrations
- Branch merges
- Infrastructure changes
- Integration of new technologies

Costs of Change

- Big changes are riskier than small changes.
- Inexpensive releases enable smaller changes.
- Smaller changes are less expensive overall.

“Hello, IT.”

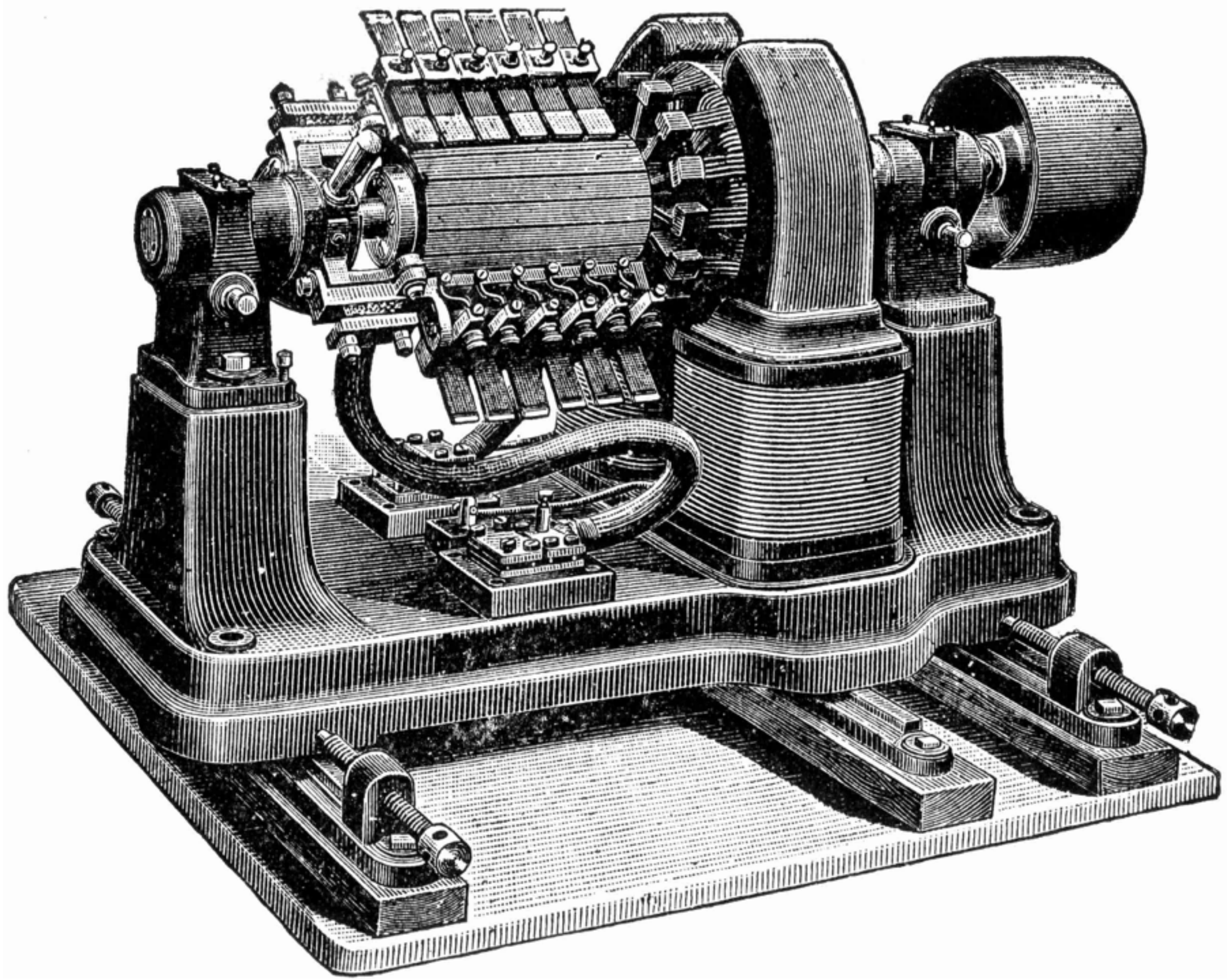
“Have you tried completely destroying the computer and replacing it with one that isn’t broken?”

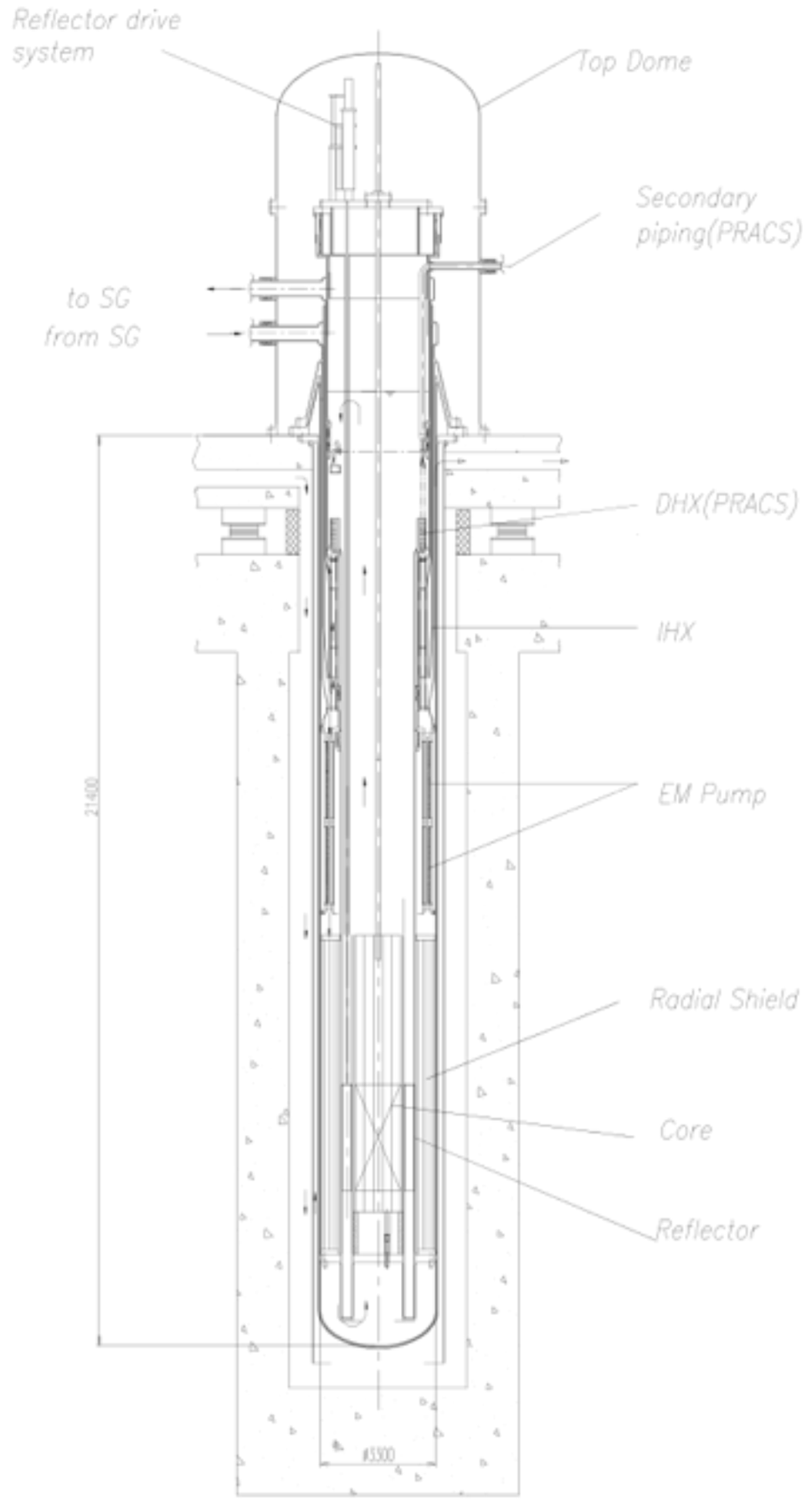
don’t fix servers

they’re free, remember?

Treat your servers
like cattle,
not pets.

Why Consume Cloud Services?




Reflector drive system
Top Dome
Secondary piping(PRACS)
to SG
from SG
DHX(PRACS)
IHX
EM Pump
Radial Shield
Core
Reflector
21400
ø5300

Heroku Postgres deals with once-per **decade** bugs **daily.**

Who has to fix it when your database develops corruption?

Amazon has
hundreds of thousands
of servers.

Who on your team is going to sleep at the co-lo?

Service-based businesses get paid when you are **happy**.

Consulting
businesses get paid
when you are **sad**.

Great services, like great consultancies, focus on making your project successful.

Opportunities

Things I’m surprised nobody has done yet.

programming environments as a service

setting up & maintaining laptop dev/staging/prod environments is the absolute worst and totally error prone

release orchestration

automating code hosting, CI, paas & monitoring
the pieces are all there!

continuous disintegration

rollbacks as a service

circuit breakers as a service

better failure modes for microservices
(because nobody anywhere ever gets this right)

Play along in the audience

Think of something horrible about development.
What if it went away? What new things could you do?

Conclusions

in which we reiterate our argument

Technology on-demand creates new opportunity.

Early usage just does
the old job better.

Real breakthroughs happen when we step back and ask:

“What next?”

fin

questions?

@pvh

Please evaluate this talk via the mobile app!

Engage

BERLIN INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

Follow us on Twitter @GOTOber