2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Advanced Usage Models for Continuous Integration in Verification Environments

John Dickol

Samsung Austin R&D Center

Agenda

- Continuous Integration
- Meet Jenkins
- Three Usage Models
- Future Work

Continuous Integration

- Software development process
- Developers check in and integrate their changes frequently
- Automatic testing of check-ins provides continuous feedback about project health

Meet Jenkins

- “An extensible open source continuous integration server”
- Easy to configure and use via web interface
- Extensions via plugins
- http://jenkins-ci.org

Jenkins Features

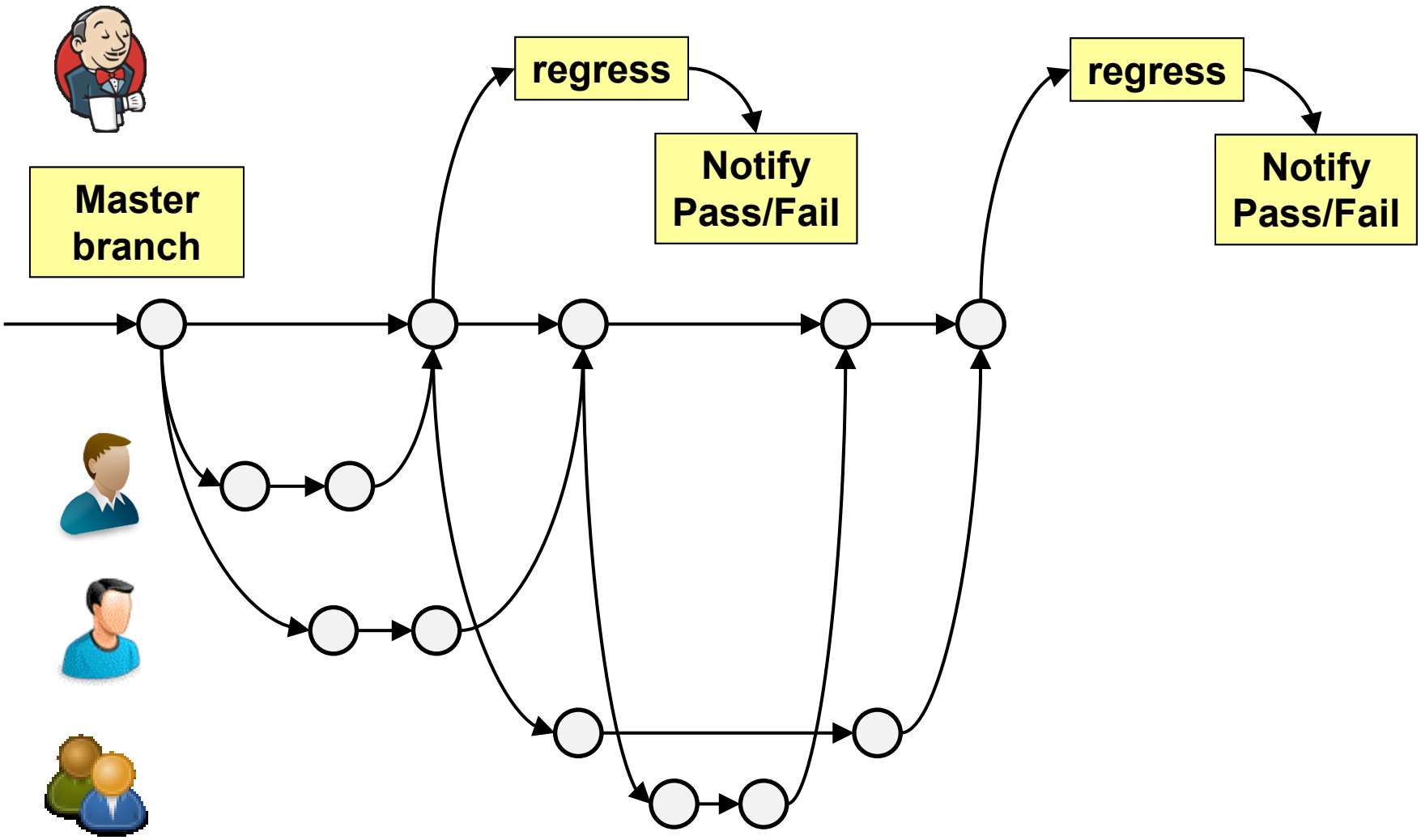
- Interacts with source control system (CVS, git, …)
  - Check for updates
  - Track changes (committer, check-in comments, etc.)
- Job scheduling
  - Time-based (like Unix cron)
  - Event-based (after code check-in, web page click, …)
- Command execution & monitoring
  - launch regression scripts
- Email notifications

Jenkins Usage Models

"Clean HEAD" Usage Model

- Built into Jenkins
- Jenkins monitors the "HEAD" of project revision database. (AKA "master" branch in git)
- Launches "smoke regression" when new check-ins are detected.
- Notifies team when regression passes/fails
- See "A 30 Minute Project Makeover Using Continuous Integration", Verilab, DVCon 2012

Clean HEAD Flow



Clean HEAD Problems

- Jenkins only informs team if model is good or bad
- Model can be broken due to single bad check-in
- Broken model can prevent users from checking in
- Cannot easily determine who broke the model if multiple check-ins regressed together.

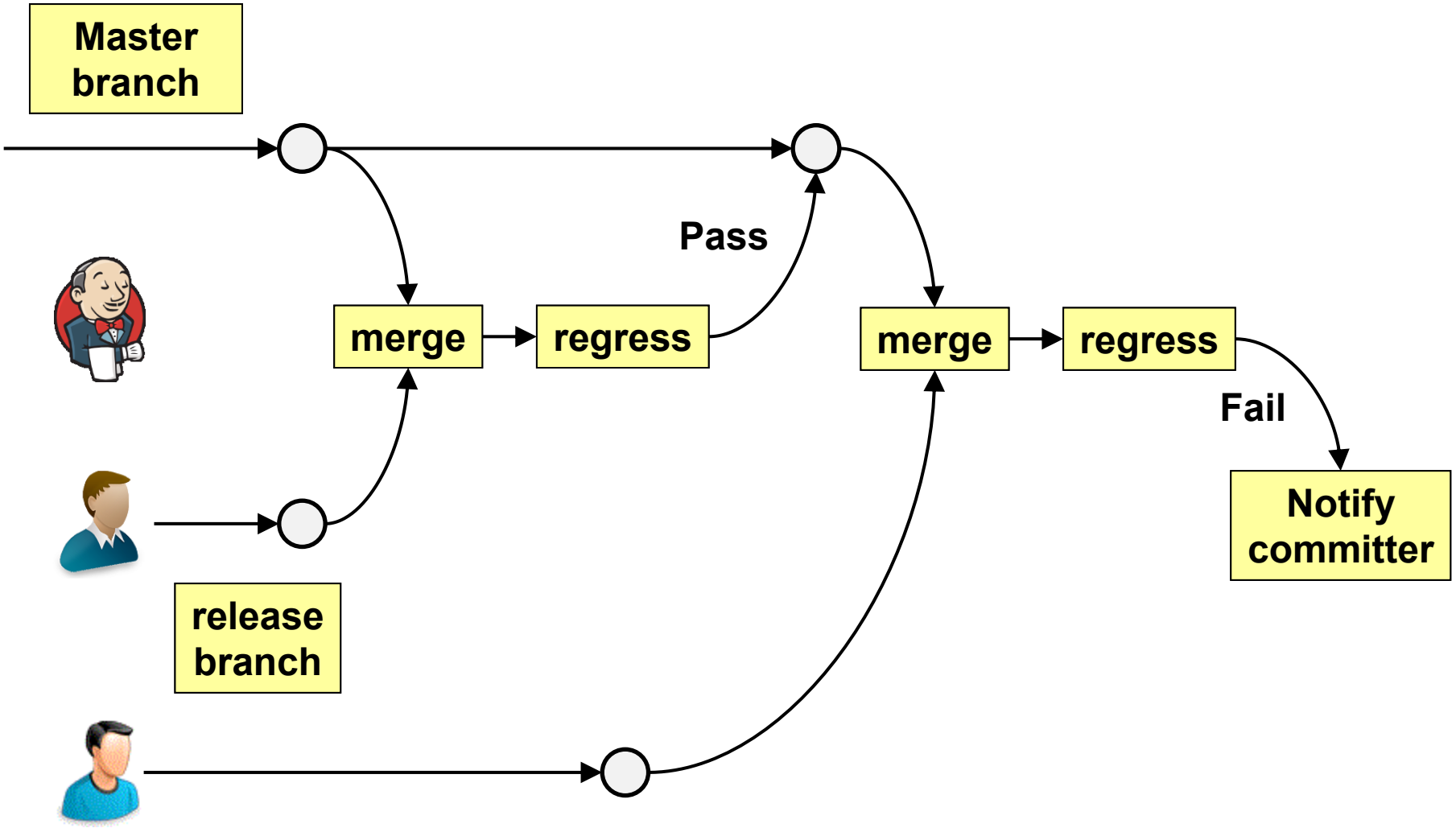
Requirements for new flow

- Maintain “known-good” version of model
- Bad check-in from one user must not break model for everyone else
- One bad check-in should not prevent others from checking in.
- Unambiguously determine who broke the model

Gated Check-in Usage Model

- Users don't update master directly
  - Check-in to individual "release branch"
  - Multiple release branches may exist
- Only Jenkins can update master branch
  - Only if check-in passes regression
- Built into Jenkins git plugin
  - Details in paper

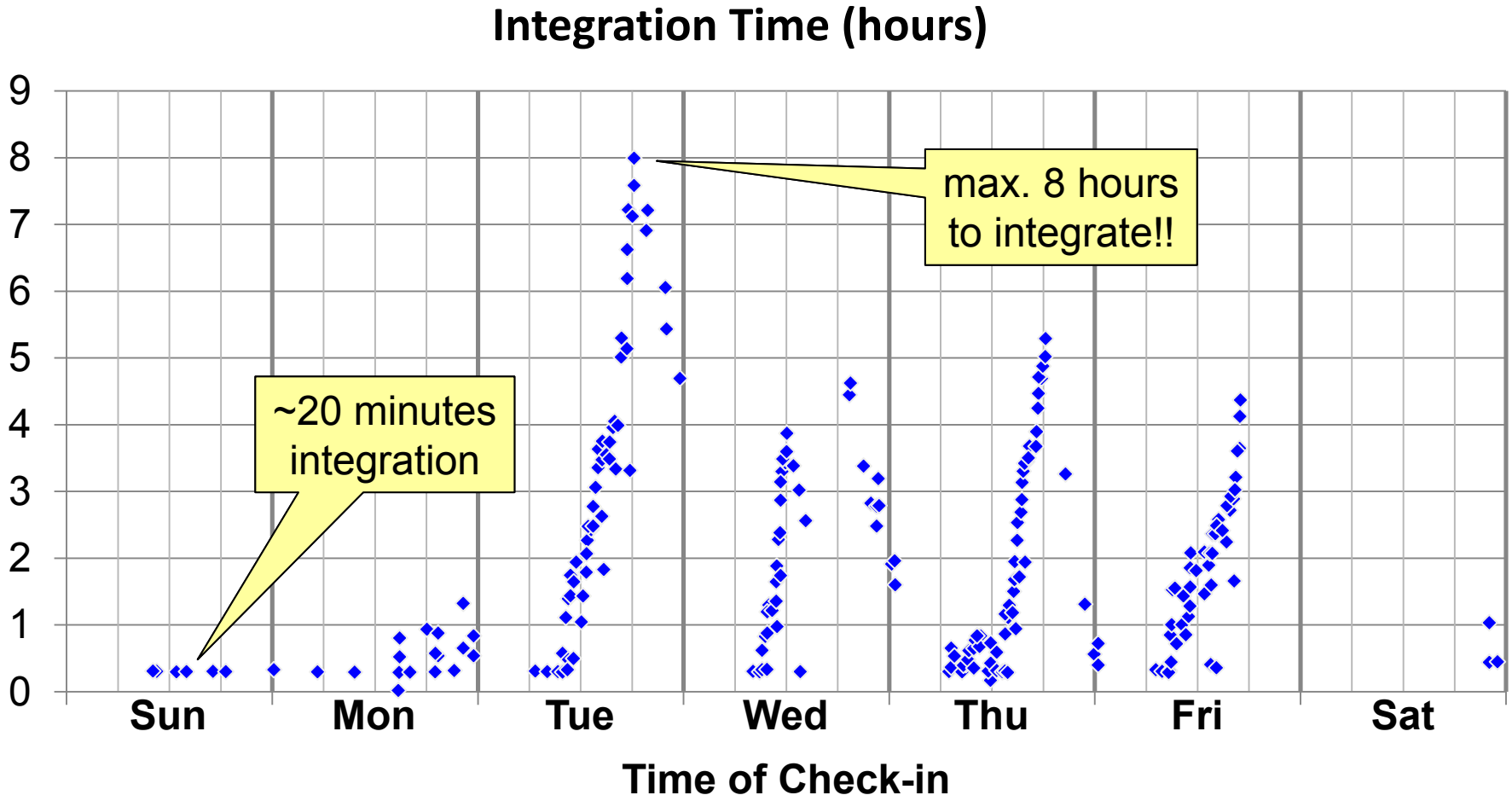
Gated Check-in Flow



Gated Check-in Results

- Master branch is always “known-good”
- Bad check-ins blocked from corrupting master
- Serialized regressions for each check-in may cause slow integration time
  - Many users check-in late in the day
  - Late check-ins may not be integrated in time for nightly regressions

Gated Check-in Integration Time



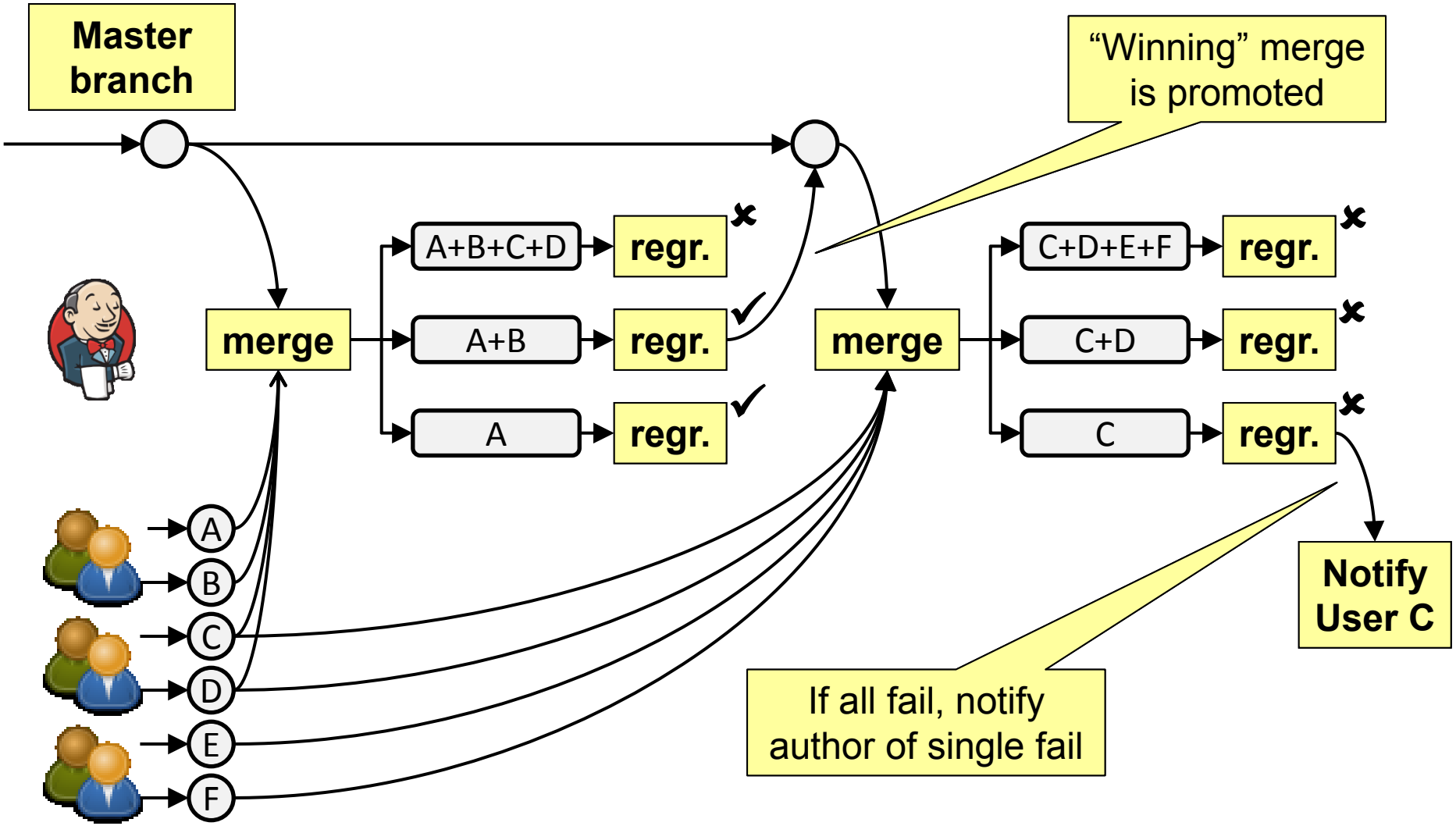
Improving Gated Check-in

- Merge & regress multiple check-ins simultaneously
- OK if merge result passes regression
- But what happens if merge result fails regression?
- Need to quickly determine which check-in is the culprit

Parallel Gated Check-in

- Jenkins launches multiple regressions in parallel
- Each regression contains merge of different number of check-ins: 1, 2, 4, …
- “Winning” regression’s check-ins get promoted to master.
- Number of parallel regressions can be configured
  - integration time vs. number of servers/licenses

Parallel Gated Check-in Flow



Parallel Gated Check-in Implementation

- Not built-into Jenkins or existing plugin

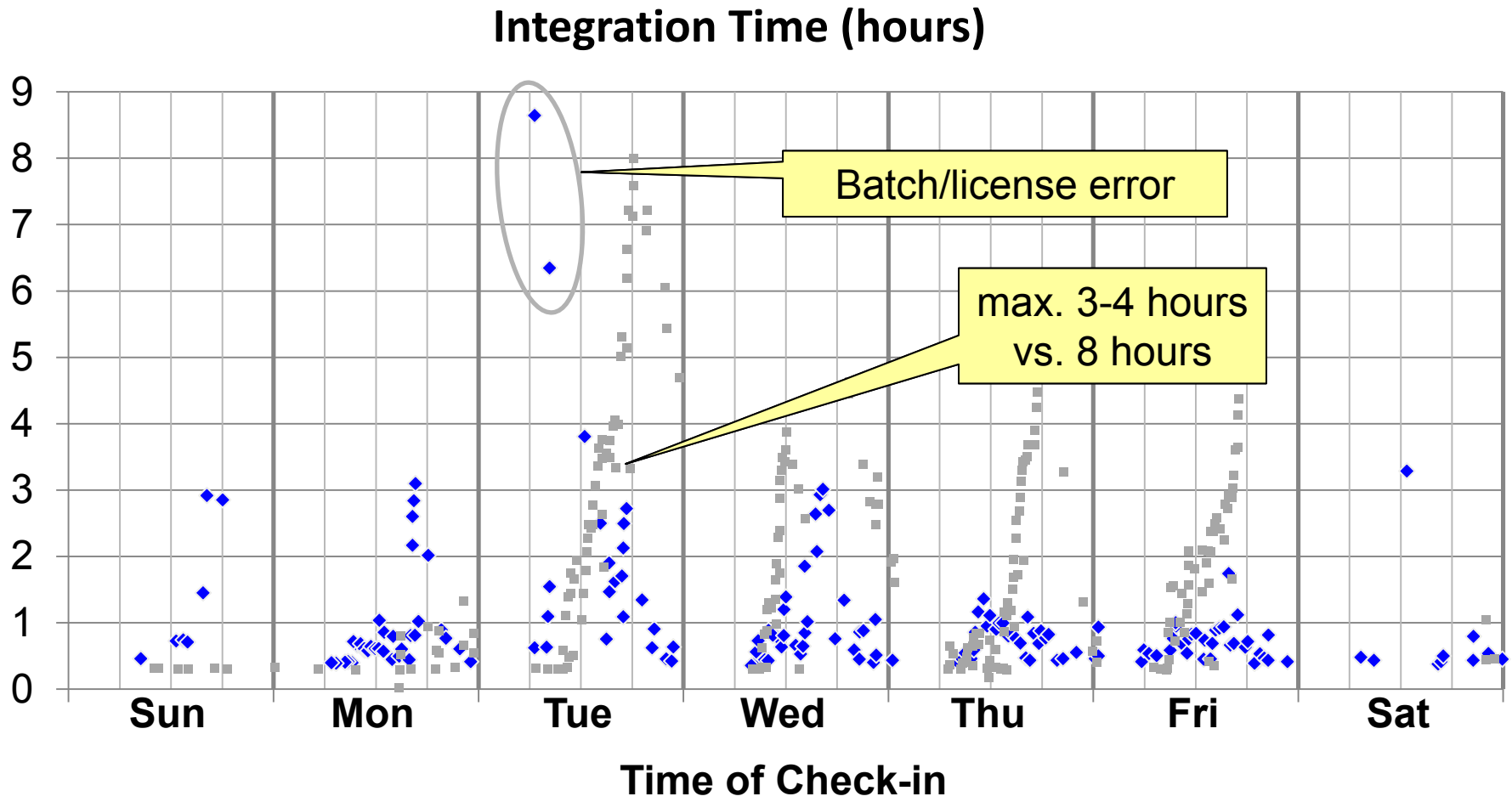
- Implemented with custom scripting on top of existing gated check-in flow:

  Plugins:
  - Parameterized Trigger
  - Conditional BuildStep
  - Groovy PostBuild

  Scripts:
  - sh
  - perl
  - groovy

- More info in the paper

Parallel Gated Check-in Integration Time (2-way)



Future Work

- Create dedicated plugin for parallel gated flow
  - Current flow is a hodgepodge of scripts, Jenkins jobs, build steps, etc.
- Investigate support for other revision control systems
  - Today git, tomorrow ???

Summary

- Clean HEAD Usage Model
  - Easy to set up
  - Good for small, well-disciplined teams
  - Tolerate some model churn
- Gated Check-in Usage Model
  - Easy to setup (for git)
  - Stable, known-good model
  - Longer integration times
- Parallel Gated Check-in Usage Model
  - More complex setup
  - Stable known-good model
  - Improved integration throughput

Thank you