Optimizing Turnaround Times In A CI Flow Using a Scheduler Implementation

Robert Strong
Contents

• Problem Statement
• Previous Flow
• Scheduler CI Flow Concept
• Implementation
• Learned Best Practices
• Effects & Results
• Future Improvements
• Questions?
Problem Statement

• Large number of changesets
• Long turnaround times
  • Accepts & Rejects
• Variable times
• Backlogs during milestones
• Slow in perfect conditions
Previous Flow

- Serial steps
  - Minimize resource usage
- Stage based
  - Large downtimes
  - Human intervention during peaks
- Incremental Improvements
  - Elimination logic
  - More testcases
- Jenkins Jobs
Previous Scenario #1
Previous Scenario #2
Scheduler CI Flow Concept

• Run testcases in parallel
  • Testcase – Unique combination of changesets
• Scheduler creates testcases
• Processor reacts to completions
• Goal
  • Turnaround time = Test time
Implementation: Individual Scheduler

- Individual scheduler grabs user request
  - In house scripts
  - Create immutable copy
  - Launch testcase
  - Ping combined scheduler
Implementation: Combined Scheduler

- Combined scheduler for testcases > 1
  - Grab state from filesystem
  - Prep workspaces
  - Launch needed testcases
  - Jenkins prop system
Implementation: Testcase State & Prop Details

• State saved in filesystem
  • Directory for active testcases
  • Multiple testcases launched/scheduler run
  • Moved to archive after completion
  • DB for pass/fail & metadata

buildFiles/16043/merge_candidate_1.prop
buildFiles/16050/merge_candidate_1.prop
buildFiles/16050/merge_candidate_2.prop
buildFiles/16055/merge_candidate_1.prop
buildFiles/16055/merge_candidate_2.prop
buildFiles/16055/merge_candidate_3.prop
buildFiles/16057/merge_candidate_1.prop

• Prop files w/all info per testcase

    CANDIDATE_JOBNUM_FILE=/nfs/disk/example_proj_integrate_parallel/buildFiles/16065/merge_candidate_1.job_num.txt
    CANDIDATE_NUM=1
    WS_SUFFIXX=81
    CANDIDATE_BRANCHES=candidate_branch1 candidate_branch2 candidate_branch3
    BUILD_TAG=jenkins-example_proj_integrate_parallel-16065
    PARENT_WORKSPACES=nfs/disk/example_proj_integrate_parallel
    PARENT_BUILDNUM=16065
    PARENT_JOBNAME=example_proj_integrate_parallel
    TRUSTED_BUNDLE=0
Implementation Details: Testcases

• Freestyle Jenkins
  • Run test suite

• Serial post-process for complete

• Non-serial for aborted
Implementation: Post-processor

• Completed mode only:
  • Changesets to push/remove
  • Kill subset & superset testcases

• Completed & Aborted mode
  • Update filesystem state
  • Recycle Workspace
  • Notify Users
Best Practices

• Lower costs allows for more testcases
  • Finite resources (batch/license/IO)

• Minimizing the cost of a testcase
  • Exit immediately on fails
  • Block external dependencies
  • Minimize false fails
  • Recycle workspaces

• “Guilt Free” launch & kill testcase
Effects & Results

• Fast & stable average & median turnaround times
  • ~15 min of overhead
  • ~50% reduction in average & median times
• Consistent results @ peak submission rate
Future Improvements

• Batch system interface
• Consolidate schedulers
• Dynamic scheduler
  • Adjust number of testcases
    • Resources
    • # of changesets
    • Pass rate
    • Priority
• Partial Testing
• Data Queries/Storage
Questions?

• Issues?
• Reliability?
• Admin Interface?