2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Optimal Usage of the Computer Farm for Regression Testing

Daniel Hansson, Verifyter

Patrik Granath, Verifyter

verifyter AUTOMATIC DEBUG

Background

Why does the need for computer farm resources increase so fast?

We run more stuff

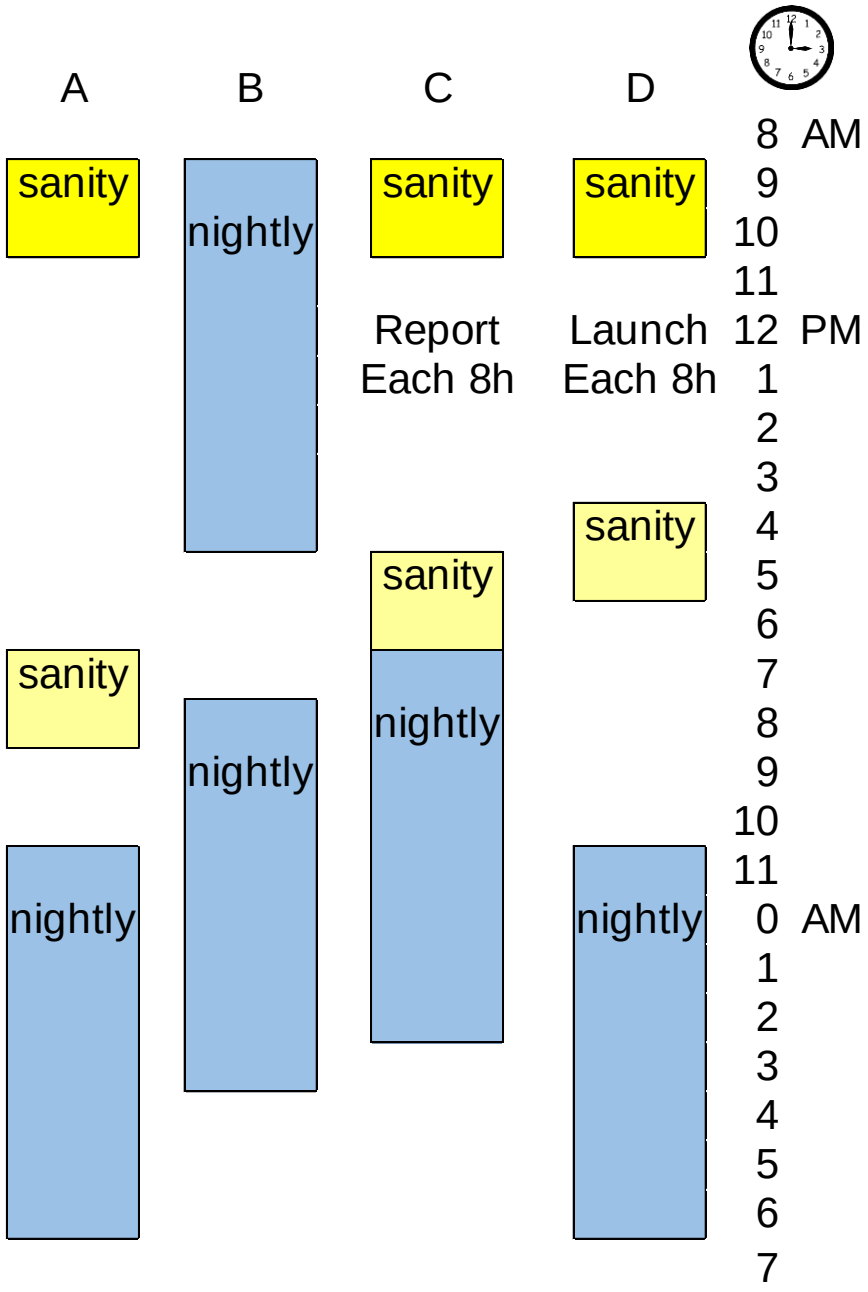
...and my jobs are still queued. Buy more!

We need some science!

Optimal Setup

Which regression test setup is the most efficient?

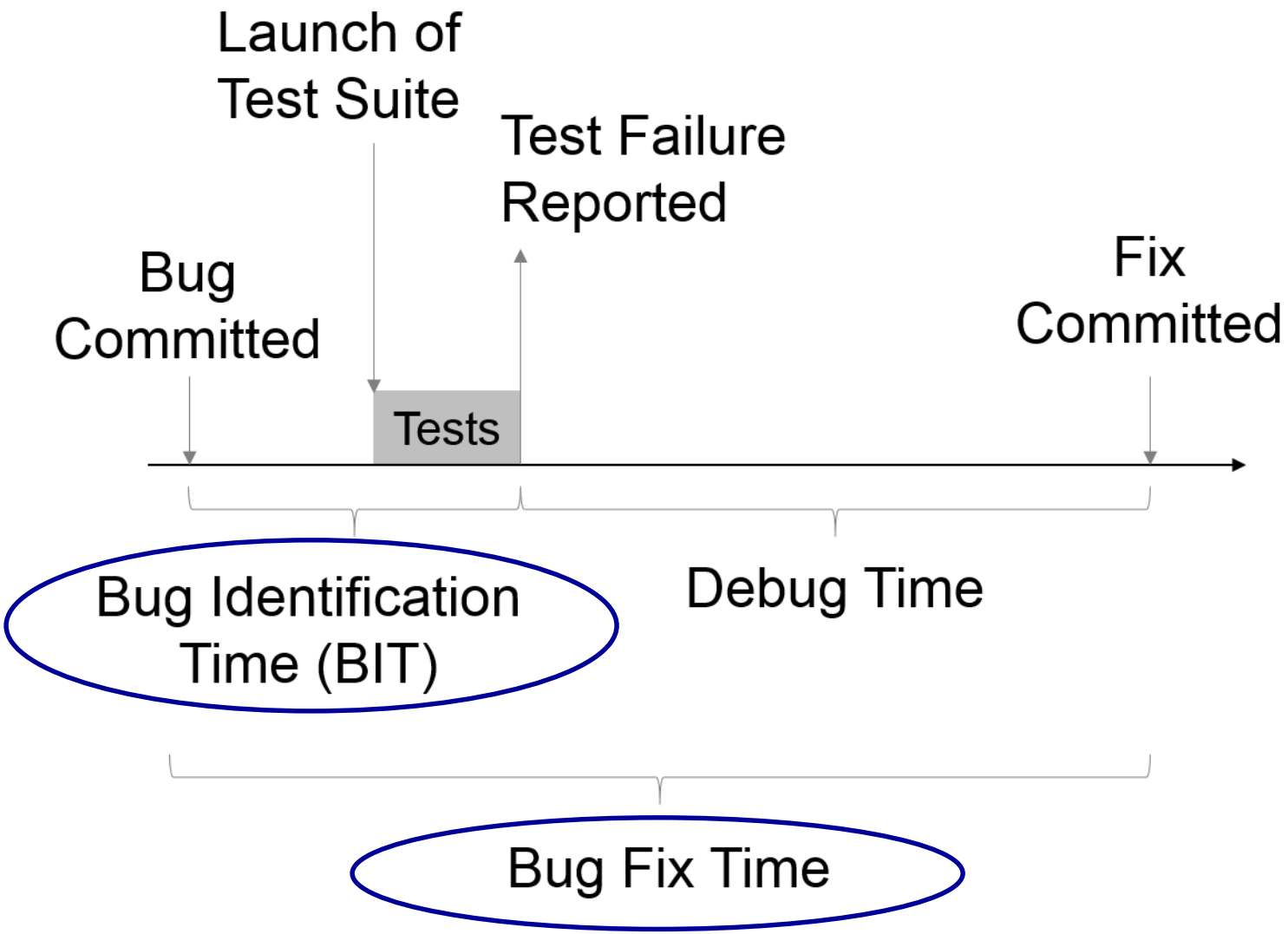
At the end of the presentation you will know



Defining Metrics (1/2)

Metric	Definition
Cost	Total CPU test time
Bug Identification Time	Time from a bug is committed to the revision control system until a failure is reported
Bug Fix Time	Bug Identification Time + debug time + committing fix
Test Fail Ratio (Quality)	The number of failing tests / total number of tests run, for a given period

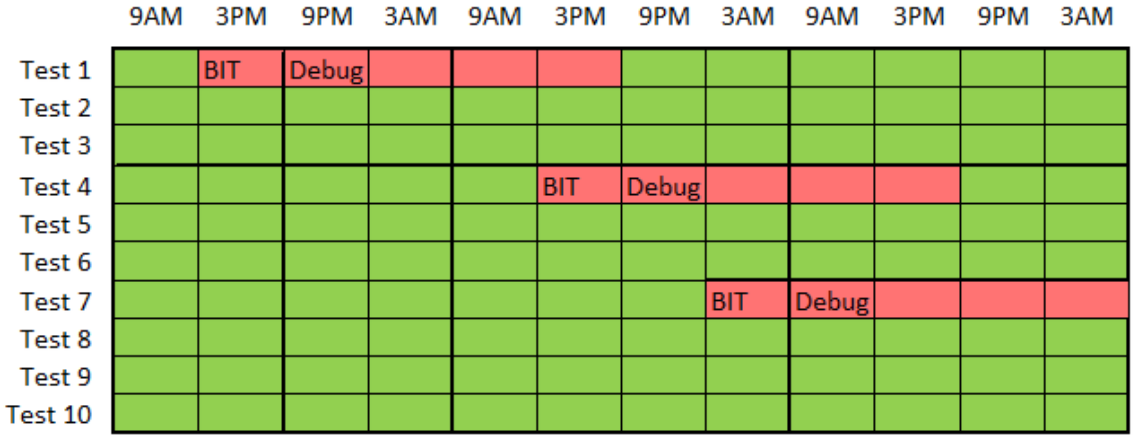
Defining Metrics (2/2)



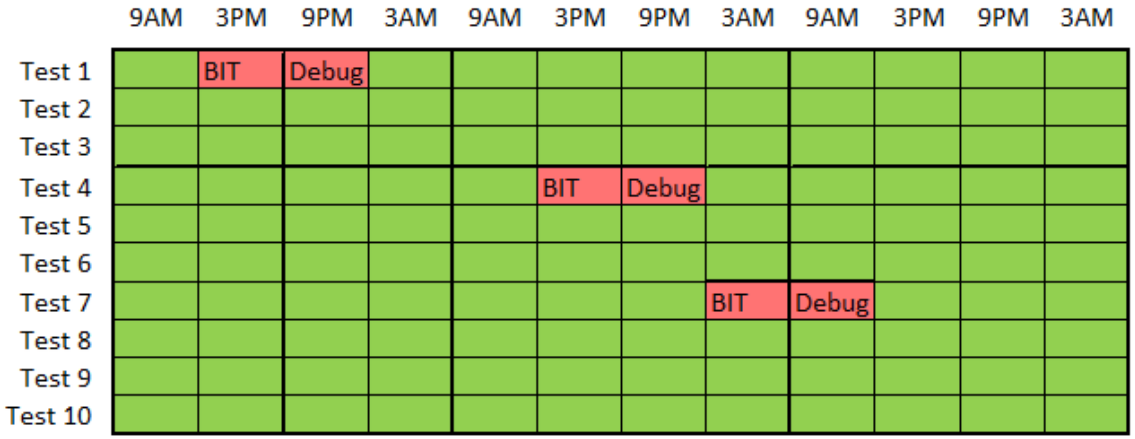
Quality

- Better Test Fail Ratio (Quality) means Earlier Release
- Shorter BIT and Shorter Bug Fix Time => Earlier Release
- What is the optimal setup?

Test Fail Ratio = 2.08% **Bug Fix Time = 30h**
Bug Identification Time (BIT) = 6h **Debug Time = 24h**

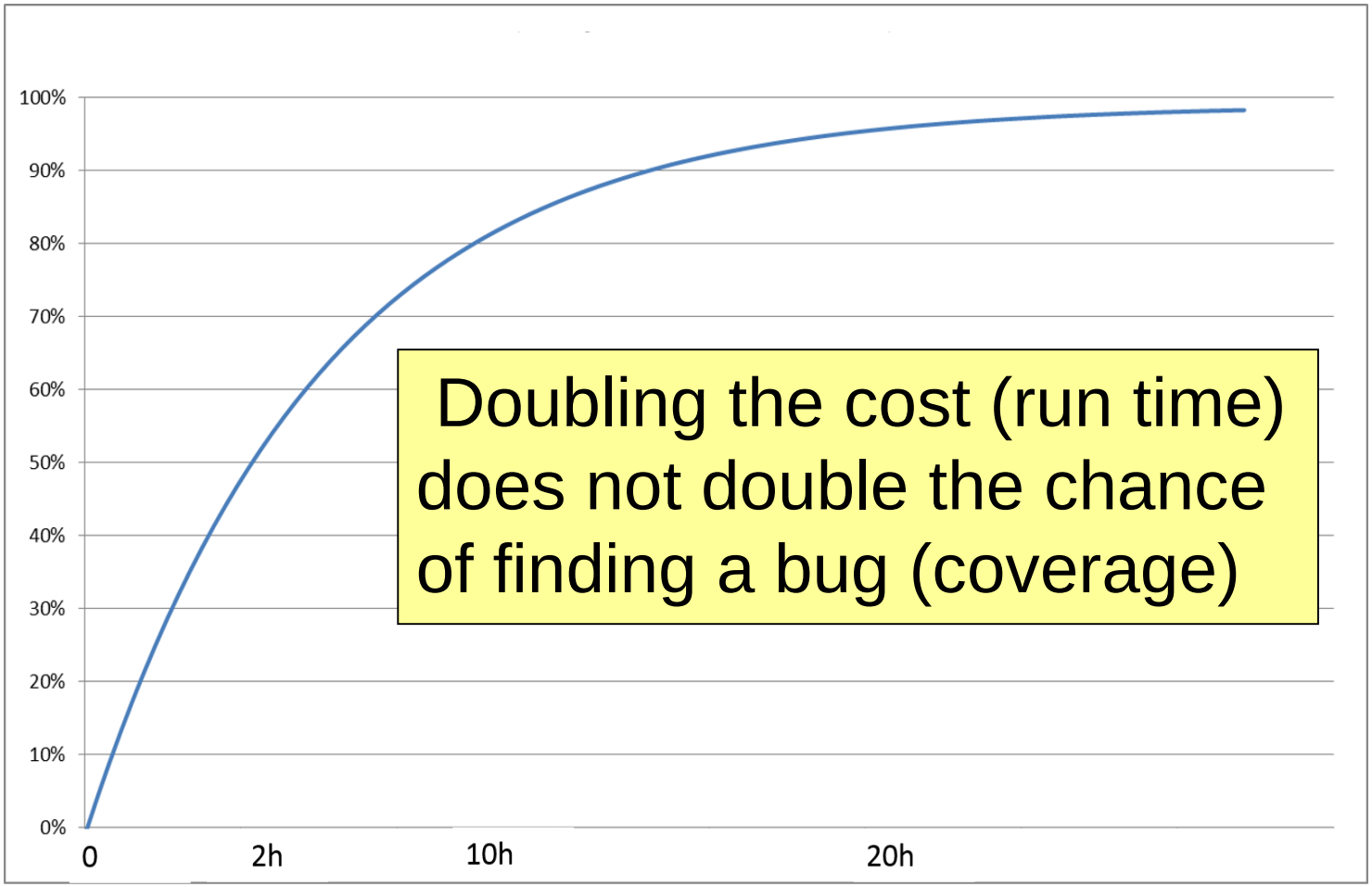


Test Fail Ratio = 0.83% **Bug Fix Time = 12h**
Bug Identification Time (BIT) = 6h **Debug Time = 6h**

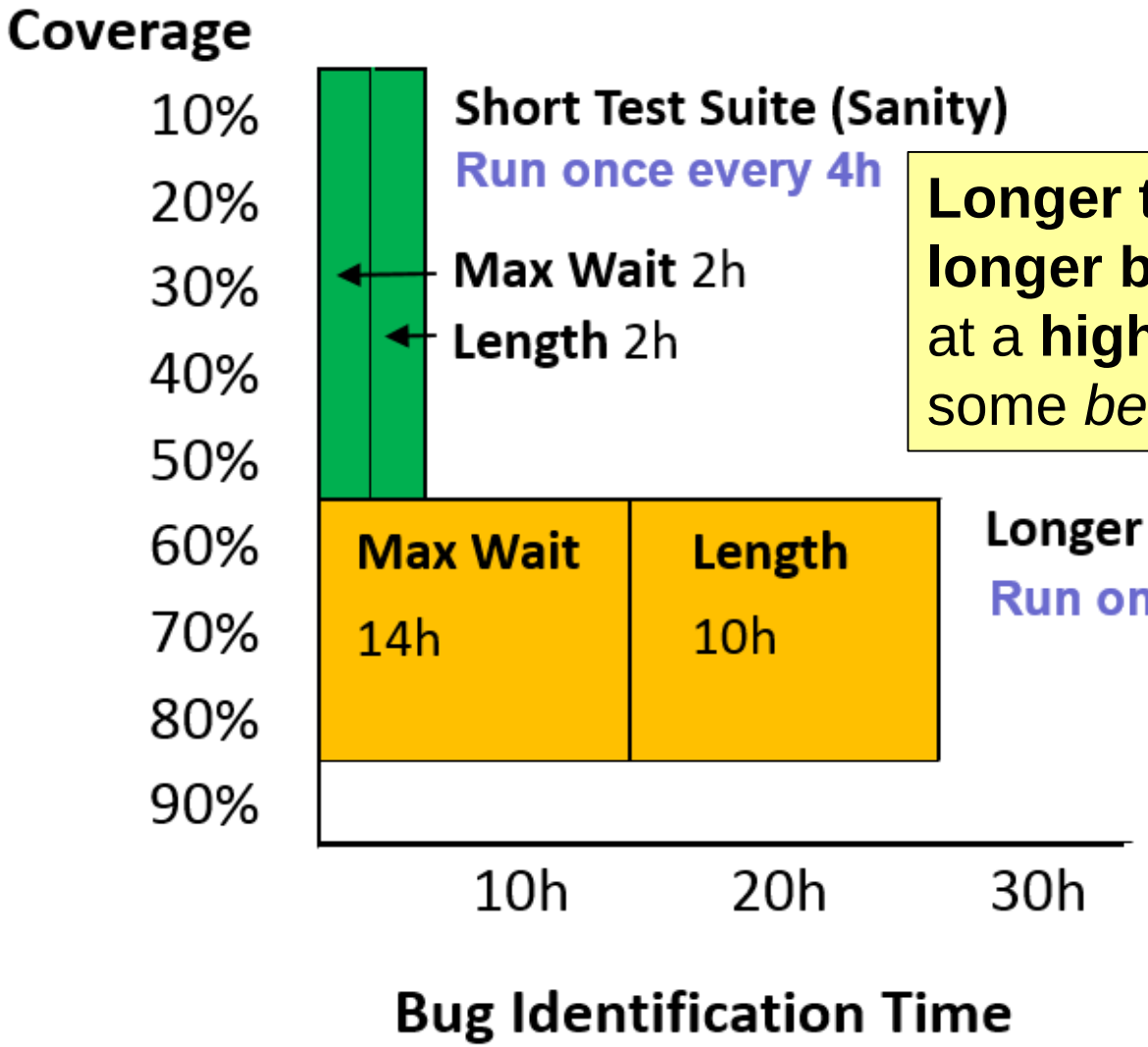


Coverage (1/2)

Functional Coverage (%) vs Test Suite Run Time (h)

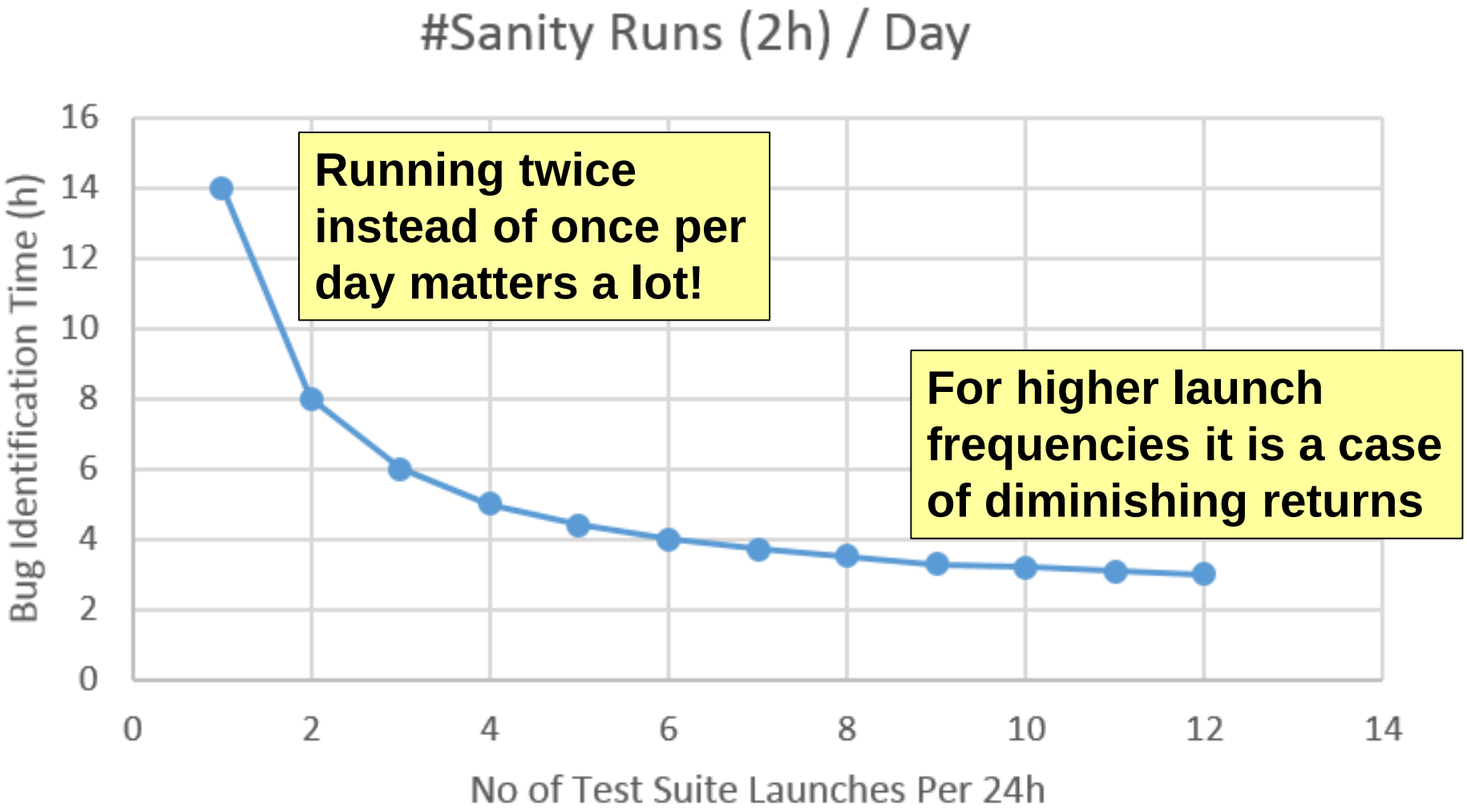


Coverage (2/2)

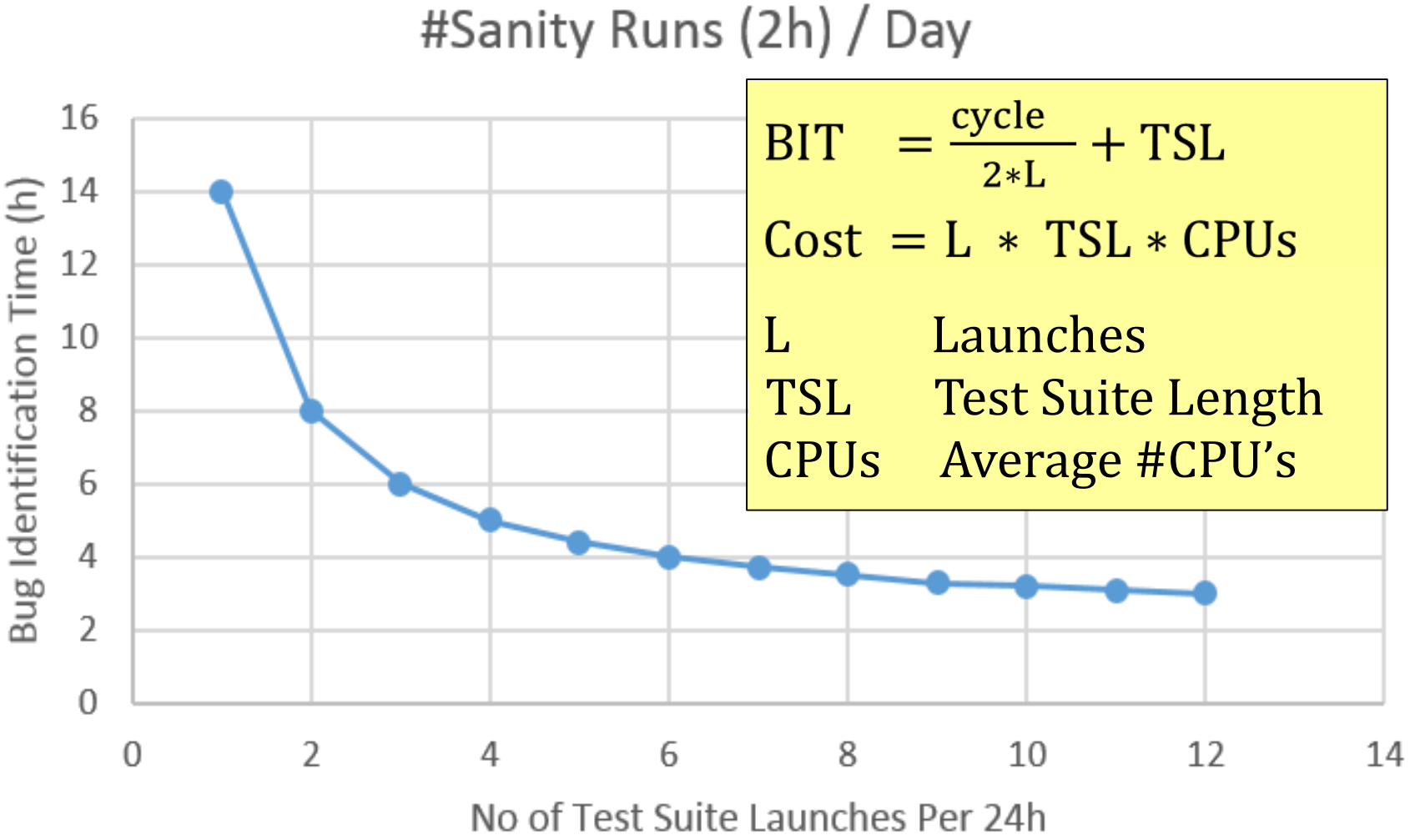


Longer test suite means **longer bug identification time** at a **higher cost**, but only gives some *better coverage*

Launch Frequency



Equations for 1 Test Suite

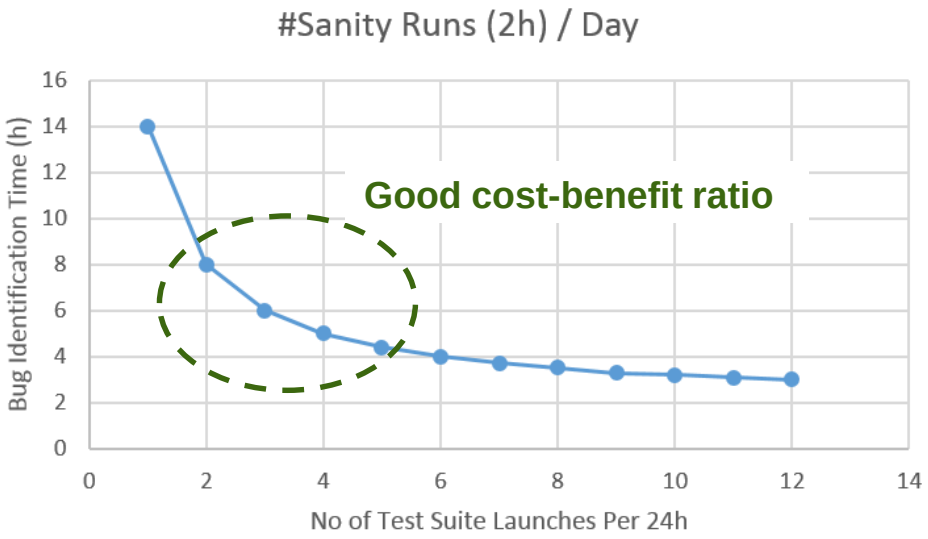


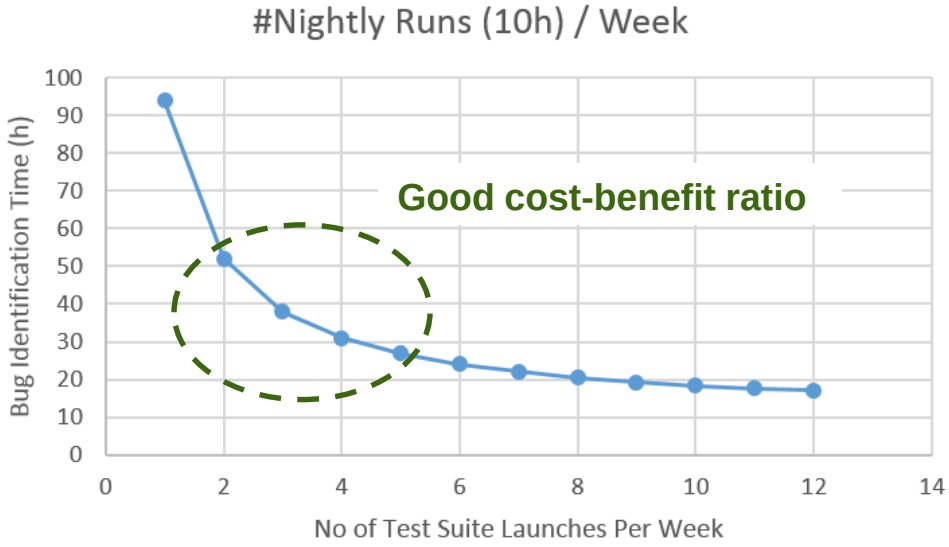
$$BIT = \frac{cycle}{2*L} + TSL$$

$$Cost = L * TSL * CPUs$$

L	Launches
TSL	Test Suite Length
CPUs	Average #CPU's

Conclusions for 1 test suite





- The BIT looks similar for different cases
- The reason: the BIT depends mainly on #launches
- **Conclusion**: 2-5 launches/cycle has a good cost-benefit ratio

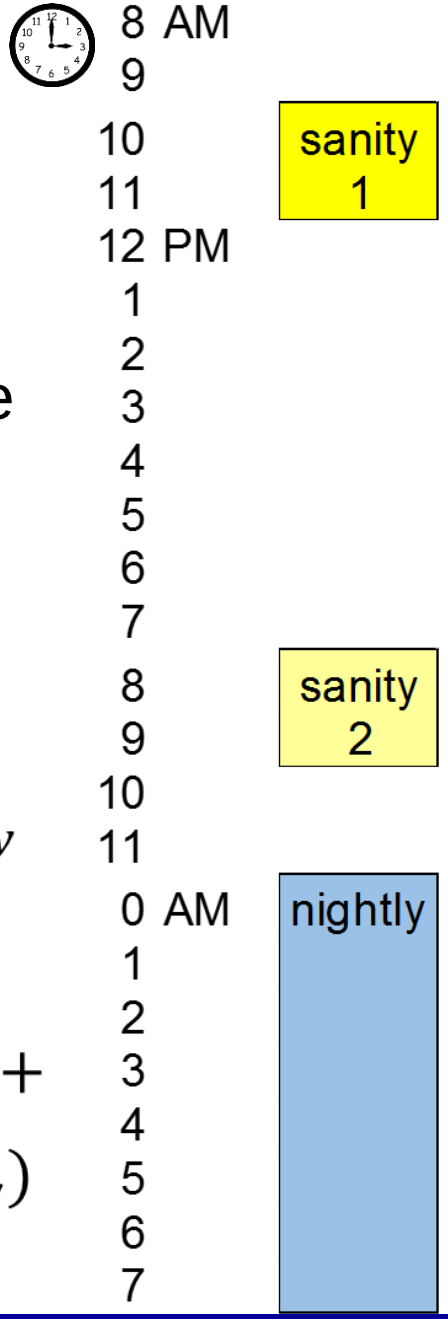
2 Test Suites

- **Assumption:** Functional coverage of *sanity* is a sub-set of the *nightly* coverage

$$\Delta coverage_{sanity} = \frac{coverage_{sanity}}{coverage_{nightly}}$$

$$\Delta coverage_{nightly} = 1 - \Delta coverage_{sanity}$$

$$\boldsymbol{BIT_{total}} = (\Delta coverage_{sanity} * BIT_{sanity}) + (\Delta coverage_{nightly} * BIT_{nightly})$$



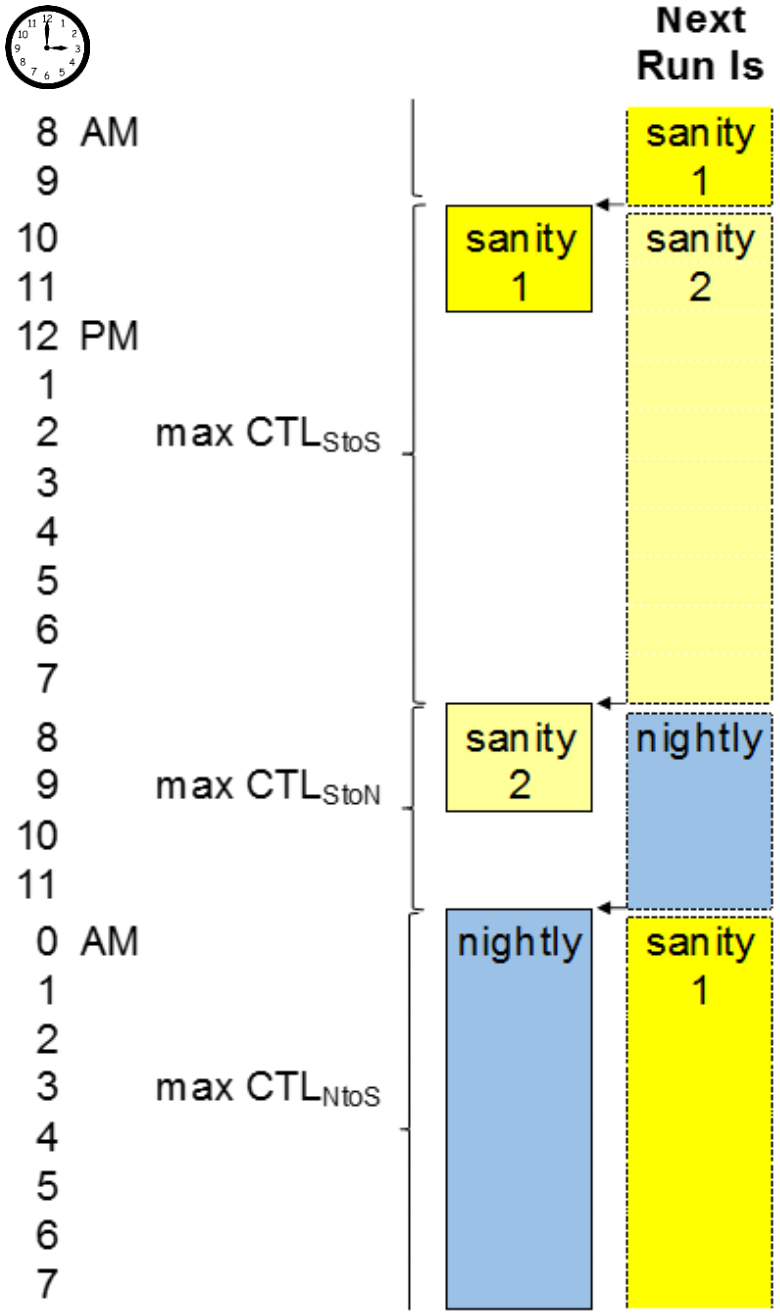
2 Test Suites

- There are 3 test suite sequences
  - *sanity => sanity*
  - *sanity => nightly*
  - *nightly => sanity*
- Probability for a bug to be committed during a test sequence

$$\frac{Max\ Commit\ To\ Launch\ (CTL)}{cycle}$$

- BIT for each test sequence

$$\frac{Max\ CTL}{2} + TSL$$



2 Test Suites

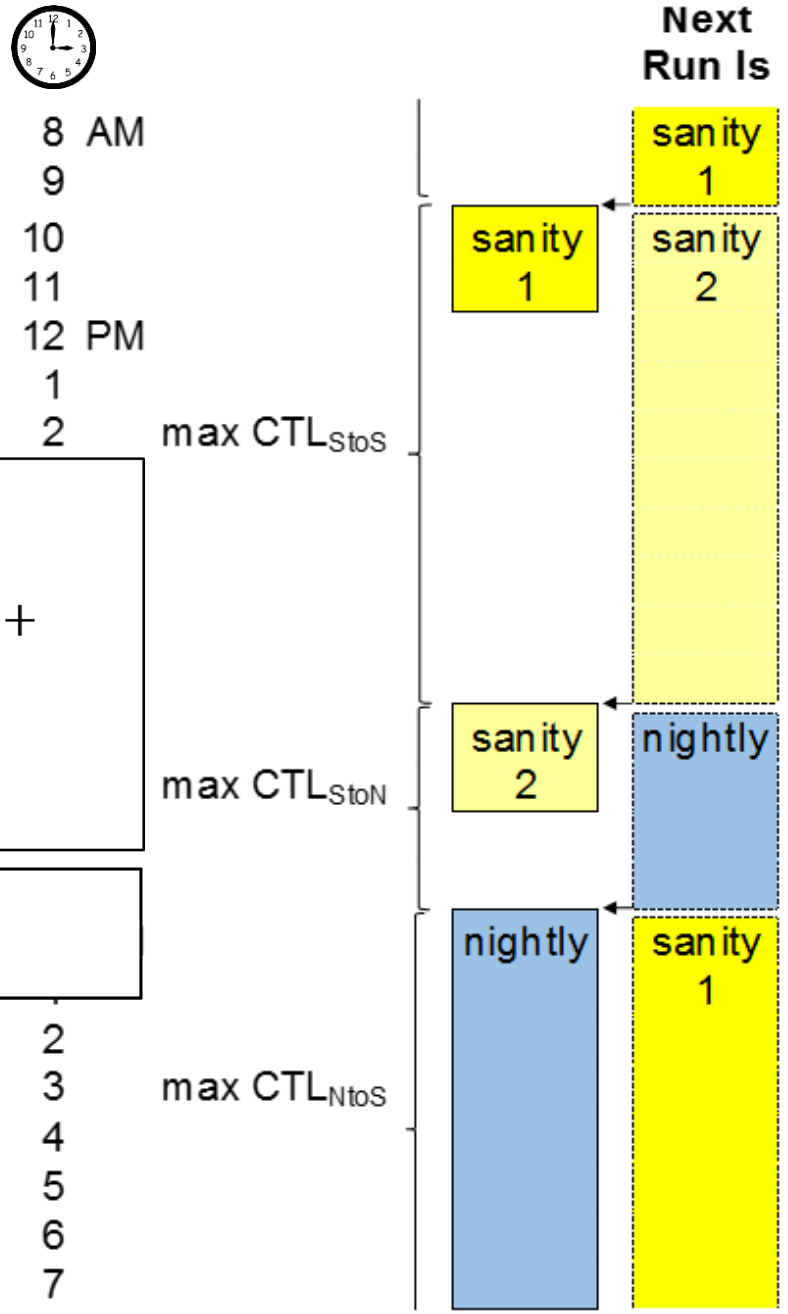
Average BIT for each of the 3 sequences

Probability for a bug to appear in this part of the cycle

$$BIT_{sanity} = \left(\frac{max\ CTL_{StoS}}{2} + TSL_{sanity}\right) * \left(\frac{max\ CTL_{StoS} * (L_{sanity} - L_{nightly})}{cycle}\right) +$$
$$\left(\frac{max\ CTL_{StoN}}{2} + TSL_{nightly}\right) * \left(\frac{max\ CTL_{StoN} * L_{nightly}}{cycle}\right) +$$
$$\left(\frac{max\ CTL_{NtoS}}{2} + TSL_{sanity}\right) * \left(\frac{max\ CTL_{NtoS} * L_{nightly}}{cycle}\right) +$$

$$BIT_{nightly} = \frac{cycle}{2 * L_{nightly}} + TSL_{nightly}$$

$$Cost_{total} = L_{sanity} * TSL_{sanity} * CPUs_{sanity} + L_{nightly} * TSL_{nightly} * CPUs_{nightly}$$



Optimal Scheduling

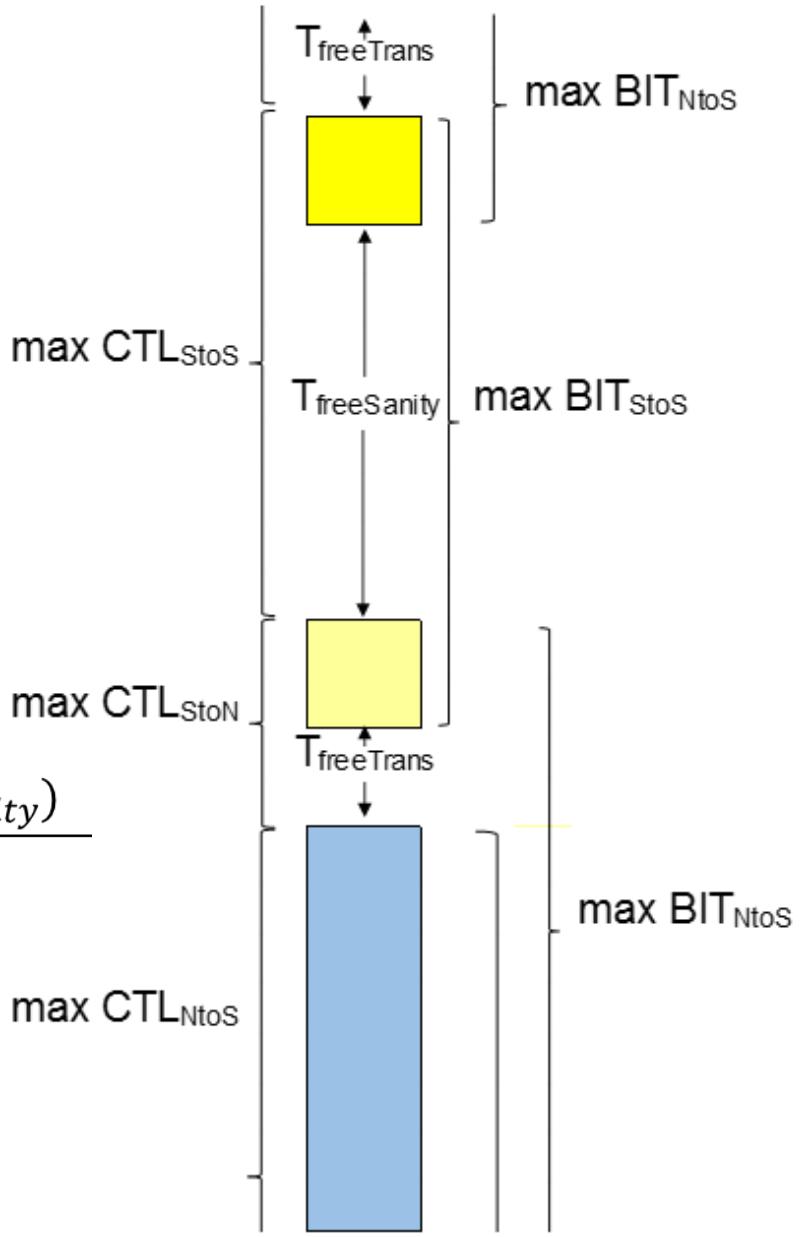
- The optimal launch schedule is achieved when all max BIT's are equal
- This keeps the max BIT's too a min

Optimal free time between two sanity runs

$$T_{freeSanity} = \frac{T_{freeTot} + (L_{tot} - L_{sanToSan}) * (TSL_{nightly} - TSL_{sanity})}{L_{tot}}$$

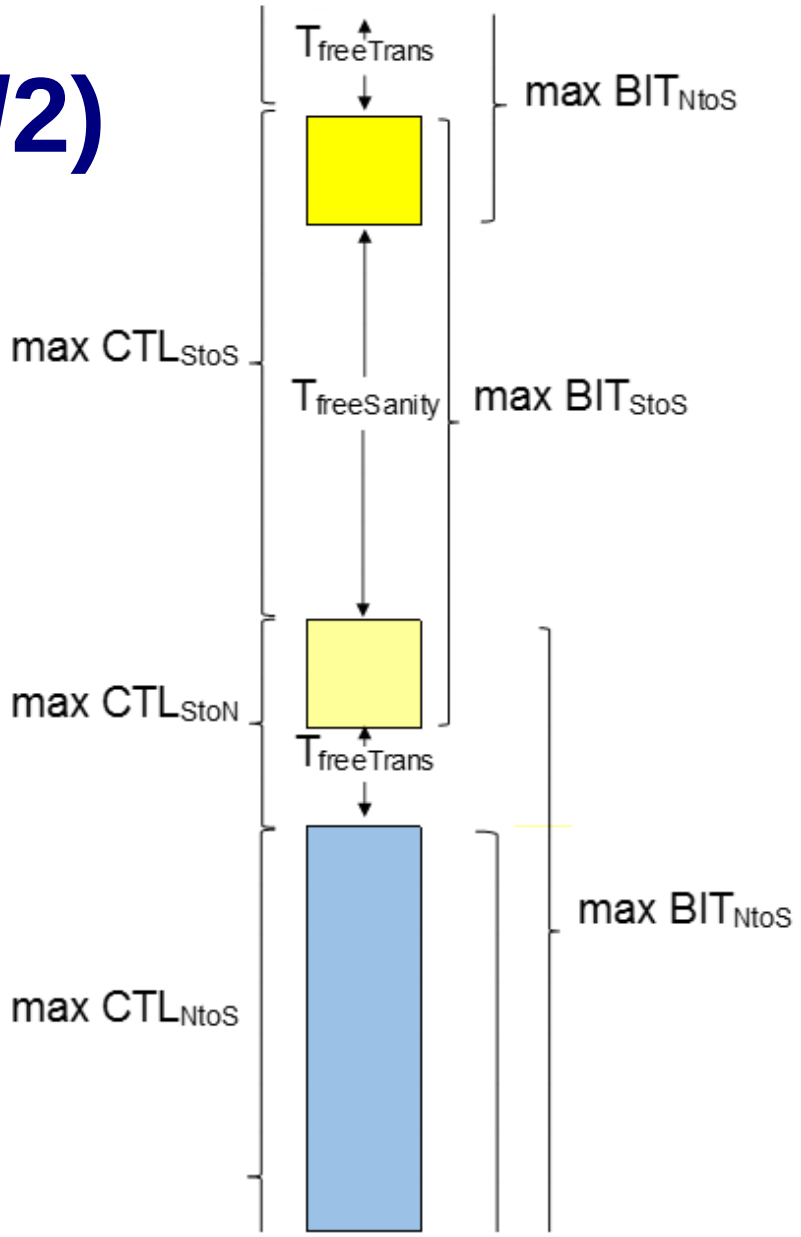
Optimal free time between one sanity run and one nightly run

$$T_{freeTrans} = \frac{T_{freeTot} - L_{sanToSan} * (TSL_{nightly} - TSL_{sanity})}{L_{tot}}$$



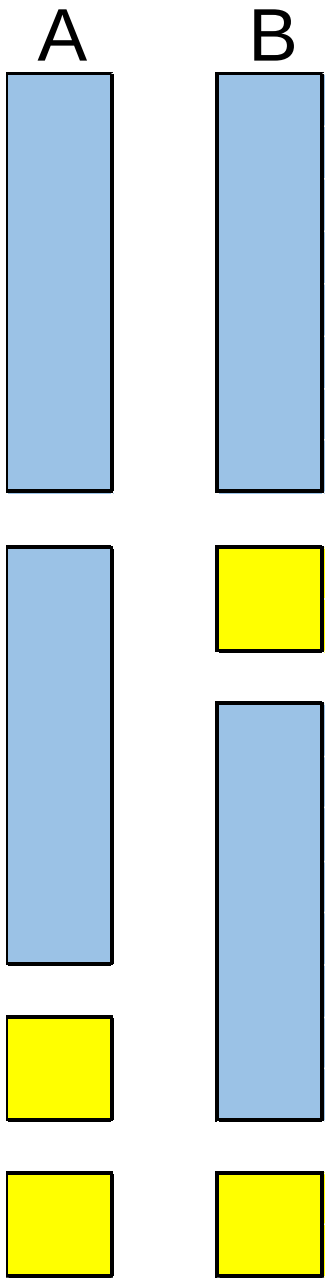
Optimal Scheduling (2/2)

- If the nightly test suite length is equal or larger than max BIT_{StoS} then nightly will never first find a bug in the sanity coverage tranche
- In this case it can be scheduled independently of the sanity runs

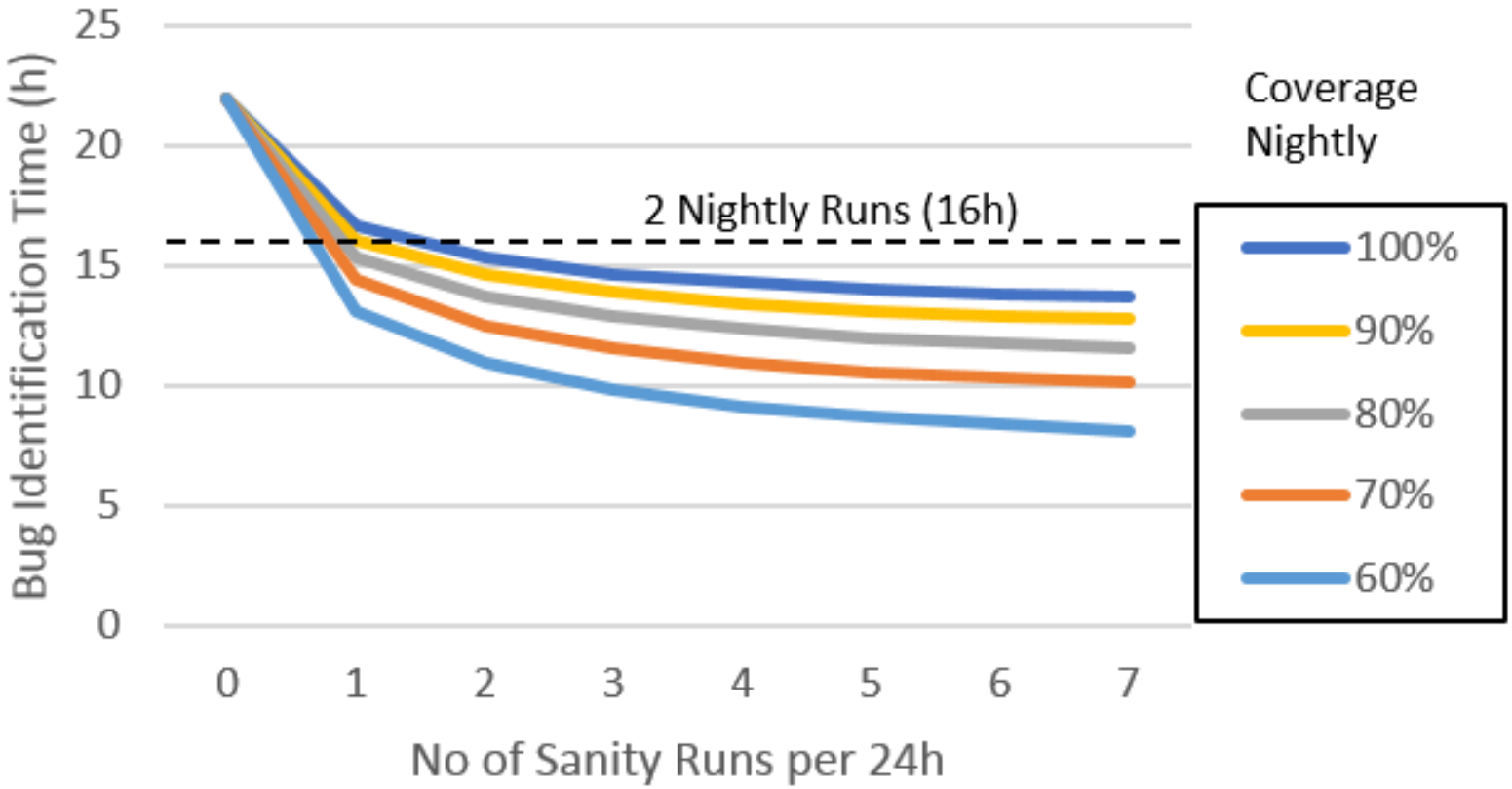


Interleaving

- Interleaving test suites reduces the overall BIT
- This is because the nightly runs will be better spread out (lower $BIT_{nightly}$)
- BIT_{sanity} is not order dependent if it is optimally scheduled
- With 3 test suites, run the shortest at least every second run



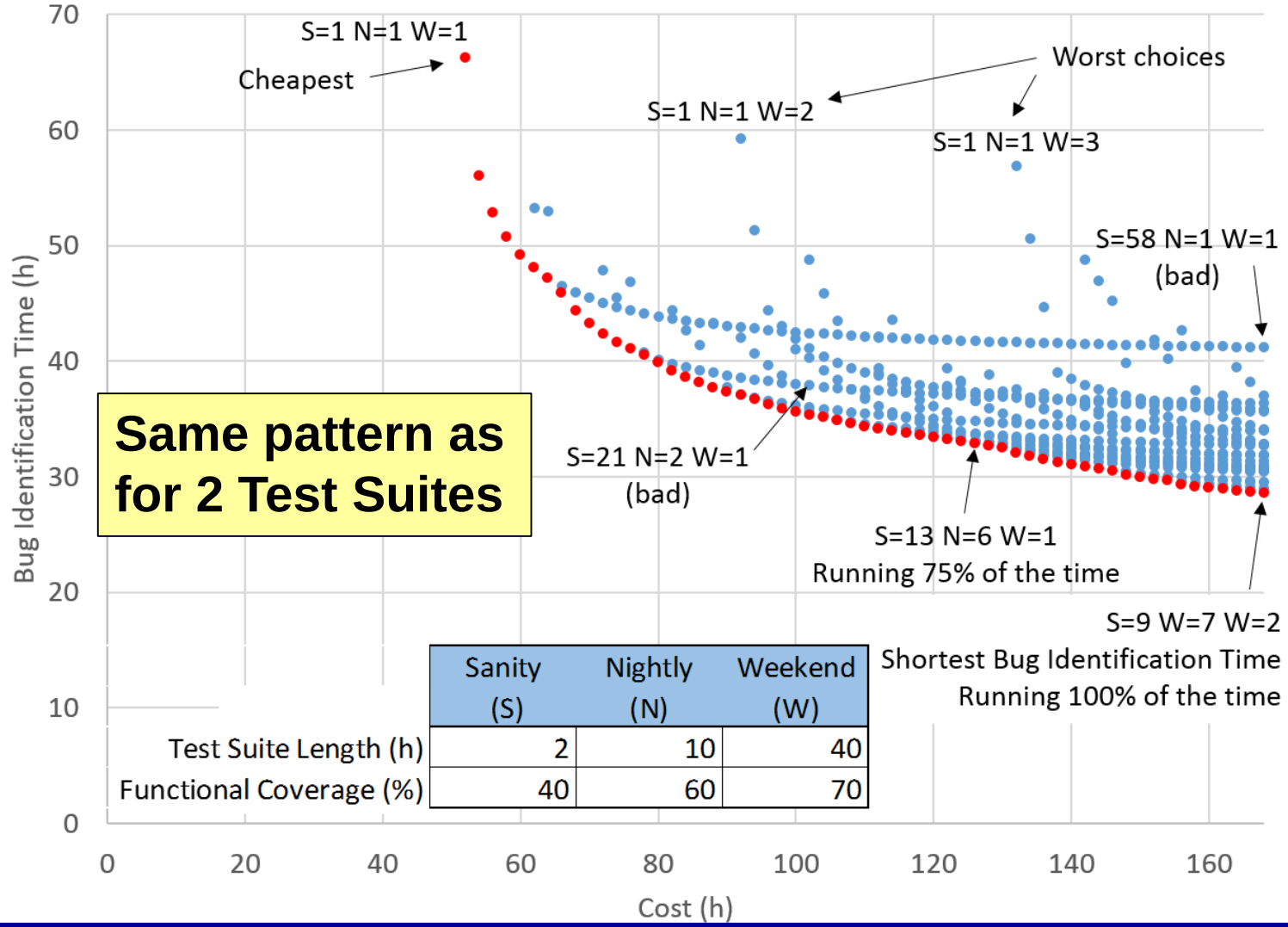
1 Nightly Run, X Sanity Runs



- 1 Nightly and 2-4 Sanity: good cost-benefit ratio
- Better than running 2 Nightly runs

3 Test Suites

Optimal Launch Frequencies (**in red**) Per Week for 3 Test Suites



Same pattern as for 2 Test Suites

	Sanity (S)	Nightly (N)	Weekend (W)
Test Suite Length (h)	2	10	40
Functional Coverage (%)	40	60	70

Summary

To find your optimal setup you need to use the equations

- Detect Bugs Fast, Fix Bugs Fast => Earlier release
- Launch Frequencies with good cost-benefit ratios:
  - 1 Test Suite: Run the test suite 2-5x per cycle
  - Multiple Test Suites: Run the largest test suite once and the smaller test suites 2-4x for each larger run
- Optimal Scheduling:
  - All max BIT's should be equal for lowest coverage tranche
  - If a test suite is longer than the max BIT of a shorter test suite then you can schedule the test suites independently
  - Interleave test suites
  - With 3 test suites, run the shortest at least every second run

Optimal Setup

Which regression test setup is the most efficient?

Answer: A