Stop Wasting Hours on Tests That Never Fail

Johannes Bechberger · @parttimenerd · SAP SE


helvetia
KARAKUN
Kanton Basel-Stadt
ERNI
INNOFUSE
+++Ba
sel++
++Te+
+++ch
Basel
basel.ch
JAVA
USER
GROUP
CH
BLKB
optravis
company of .msg
cyon
CS2
CSS
IRIX
swiss made
software
ICT
Scouts
Wir fördern Zukunft
SD:>_
[SwissDevJobs.ch]
#BaselOne26
baselone.ch

DEMO D0

Running tests takes time

```
cd third-party/commons-collections
./make-change-commons.sh # remove one line from remove()
mvn test
```

> *"Very few of our tests ever fail, but those that do are generally 'closer' to the code they test."*

Memon et al., Taming Google-Scale Continuous Testing, ICSE-SEIP 2017

If a test hasn't called
the changed code,
it cannot fail on this change.

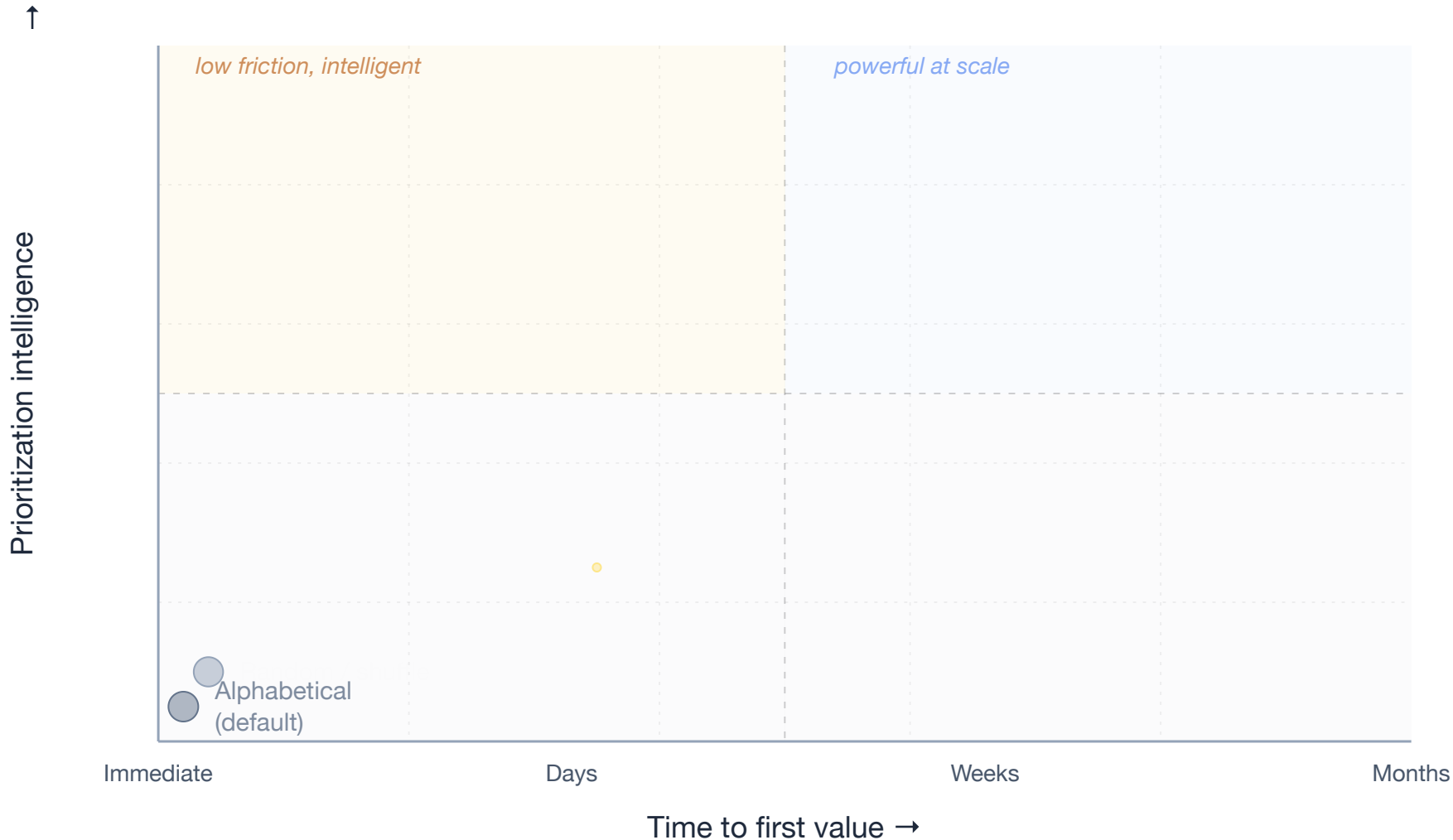
There's a zero-config, local solution.

test-order

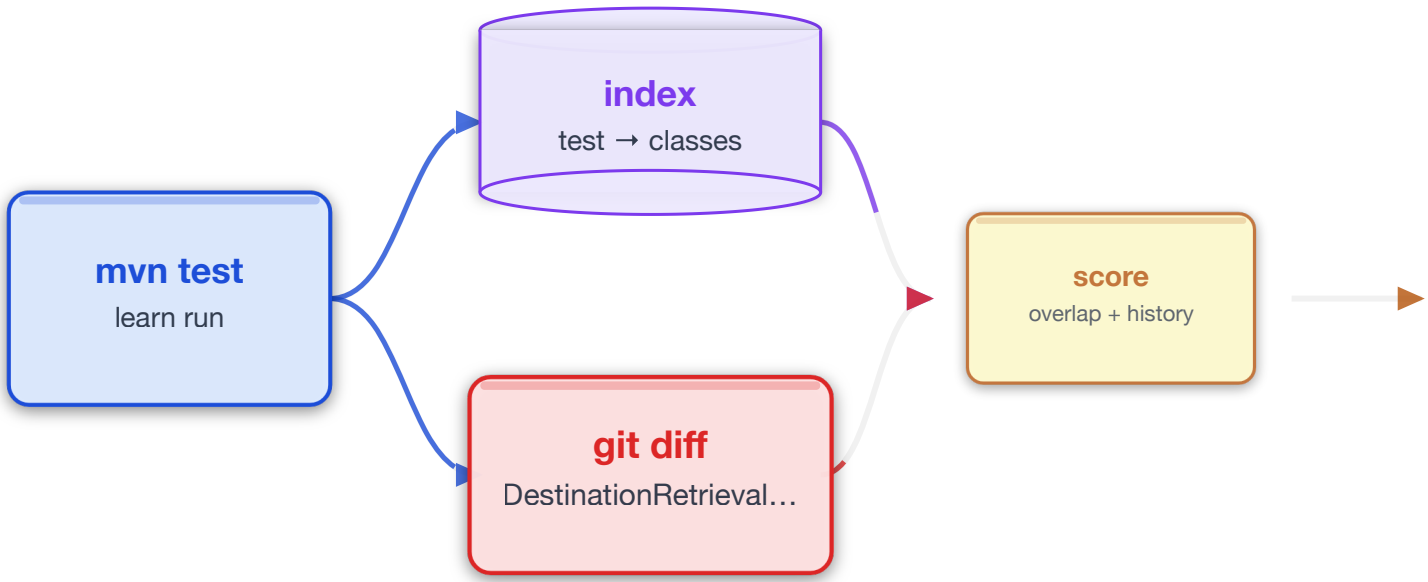




Failure found in first 10% of the suite vs. 80% — same tests, different order.


↑
low friction, intelligent
powerful at scale
Prioritization intelligence
Alphabetical
(default)
Immediate
Days
Weeks
Months
Time to first value →

Learn once. Rank on every run.



DEMO D1

Same bug. Test jumps from #237 to #1.

```
cd third-party/commons-collections
./setup-test-order.sh   # adds .mvn/extensions.xml
                        # and restores index
./make-change-commons.sh
mvn test-order:show
mvn test
```

DEMO D2

Real project. Live edit. Wrong test first → right test first.

```
cd third-party/spring-petclinic
mvn test                          # learn run — records dep map
./petclinic-make-change.sh        # edit VisitController.java
mvn test-order:show
mvn test                          # VisitControllerTests → #1
```

Step 1: add to `pom.xml`

```
<plugin>
  <groupId>me.bechberger</groupId>
  <artifactId>test-order-maven-plugin</artifactId>
  <version>0.1.0</version>
  <extensions>true</extensions>
  <executions><execution>
    <goals>
      <goal>prepare</goal>
    </goals>
  </execution></executions>
</plugin>
```

Step 2: multi-module? Add `.mvn/extensions.xml`

```xml
<?xml version="1.0" encoding="UTF-8"?>
<extensions>
  <extension>
    <groupId>me.bechberger</groupId>
    <artifactId>test-order-maven-plugin</artifactId>
    <version>0.1.0</version>
  </extension>
</extensions>
```

📦 shared index across modules 🔍 `test-order:show` · `test-order:affected`

Step 3: `mvn test` auto-detects

① **No index found → learn run**

Bytecode agent on · records every method call · writes `.test-order/` · normal test results

② **Index found → order run**

Agent off · loads index · intersects with `git diff` · reorders tests · zero overhead

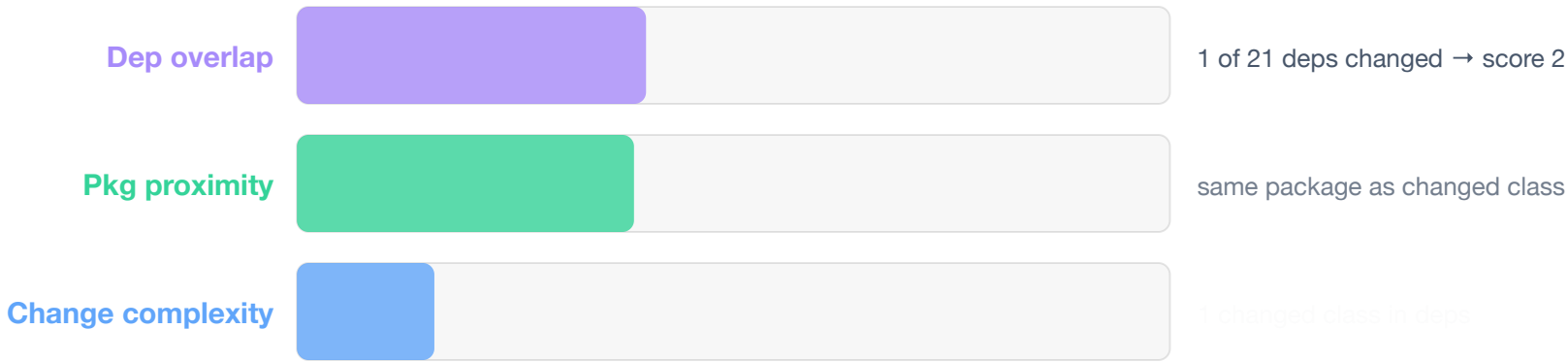
Commit or CI-cache `.test-order/` to skip the learn run on CI.

Gradle? Three lines.

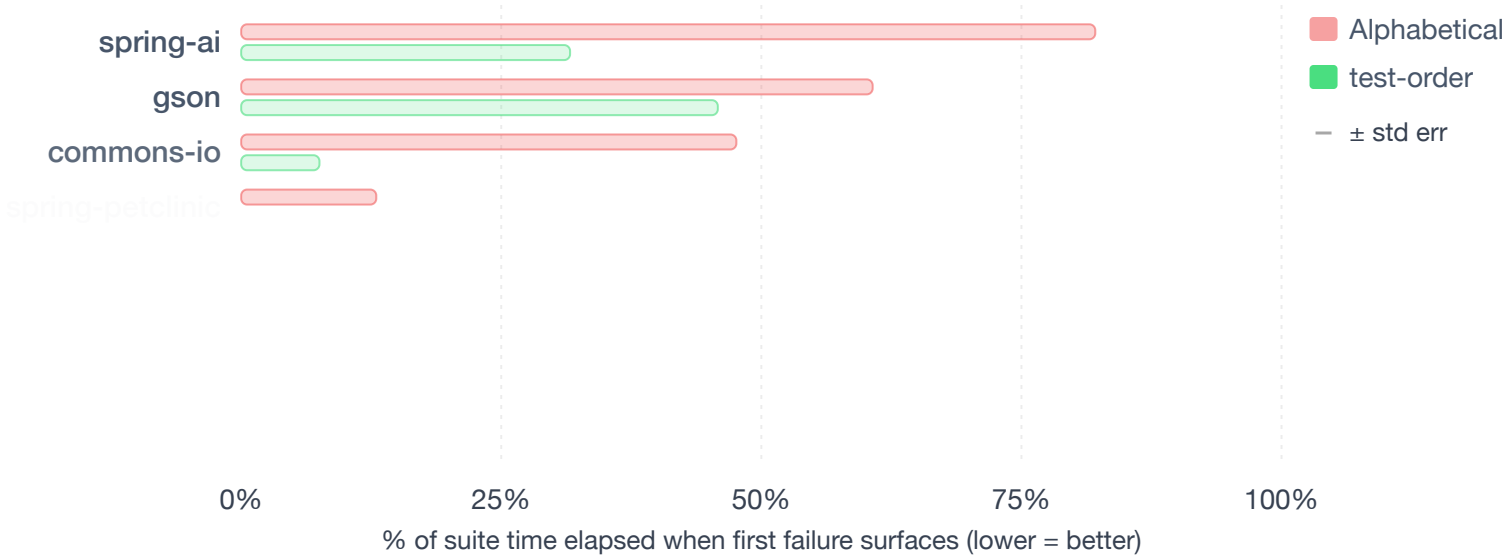
```
plugins {
    id 'me.bechberger.test-order' version '0.1.0'
}
```

```
./gradlew test   # same index format, same dep map
```

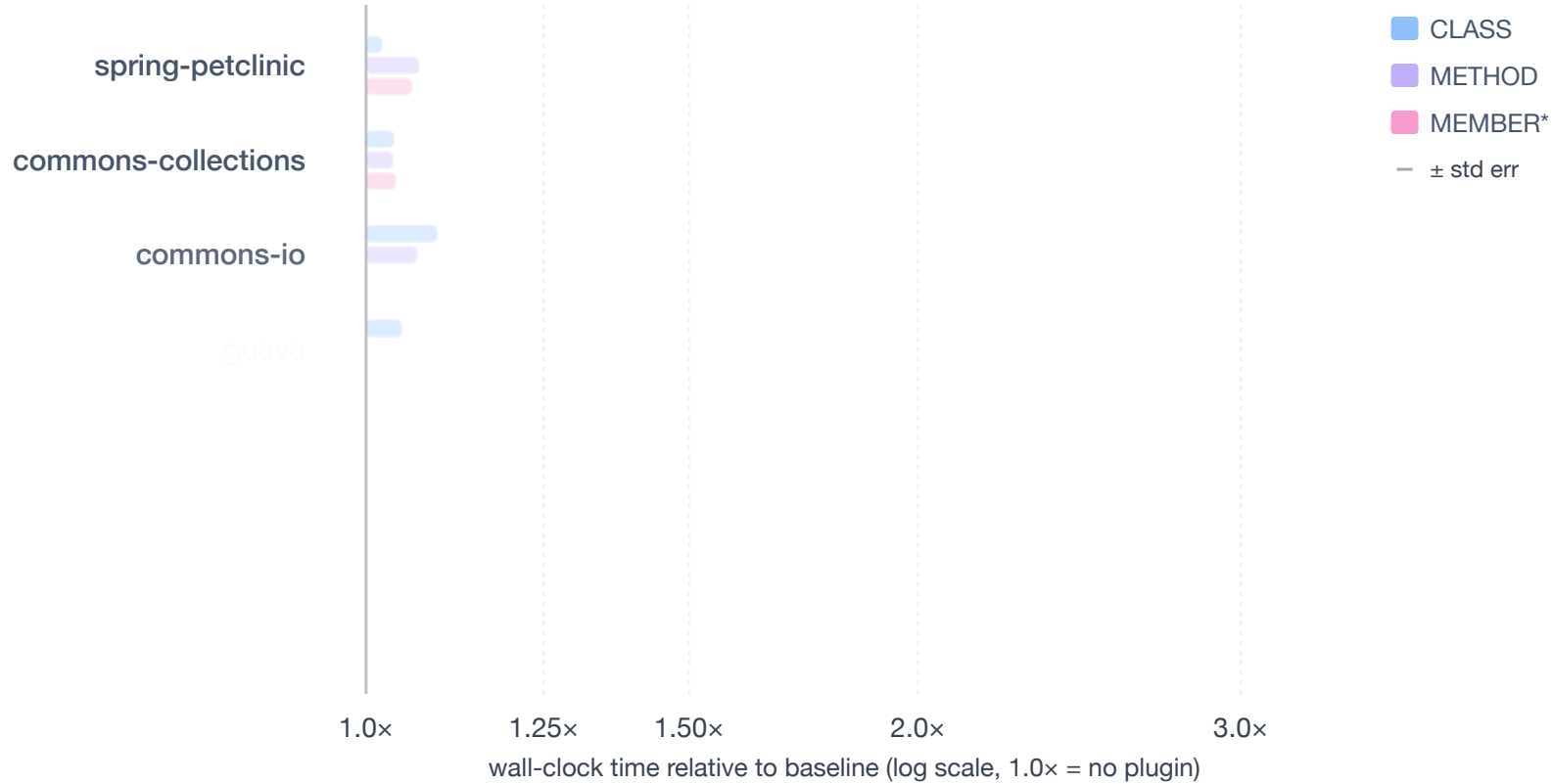
Why did VisitControllerTests rank #1?



When does the first test fail?



The cost: one learn run. Zero overhead after that.



Let's go into the dashboard.

DEMO D4

test-order:dashboard

```
cd third-party/spring-petclinic
mvn test-order:dashboard
open target/test-order-dashboard/index.html
```

Six numbers that tell the story

APFD

Average % of faults detected early

Failures

Tests that failed in recent runs

Pass streak

Consecutive green runs per test

At-risk

Tests showing DEGRADING trend

Time saved

Minutes not spent on wrong tests

Health A–F

Overall suite quality grade

GitHub Actions — drop-in config (1/2)

```yaml
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
        with: { fetch-depth: 0 }        # needed for git diff

      - name: Restore test-order index
        uses: actions/cache@v4
        with:
          path: .test-order/
          key: test-order-${{ github.ref }}-${{ github.sha }}
          restore-keys: |
            test-order-refs/heads/main-
            test-order-

      - name: "Learn: full instrumented run"
        if: github.ref == 'refs/heads/main'
        run: mvn test -Dtestorder.mode=learn
```

GitHub Actions — drop-in config (2/2)

```yaml
      - name: Run affected tests
        if: github.ref != 'refs/heads/main'
        run: |
          mvn test-order:affected test \
            -Dtestorder.changeMode=since-last-commit

      - name: Run remaining tests (if affected passed)
        if: github.ref != 'refs/heads/main'
        run: mvn test-order:run-remaining test

      - name: Save test-order index
        if: github.ref == 'refs/heads/main'
        uses: actions/cache/save@v4
        with:
          path: .test-order/
          key: test-order-${{ github.ref }}-${{ github.sha }}
```

See it live

github.com/parttimenerd/sample-ci-test-order

→ Actions tab



Large multi-module project · 52 modules

90s

`mvn clean test`

Compile crawl. Zero tests started.

55s → RED

`mvn test-order:affected test`

Right failure. Right away.

80% earlier signal × 30 pushes/day = a completely different dev loop.

Fine-grained control

```
@AlwaysRun           // always ranked #1 regardless of score
class SmokeTest { }

@TestOrder(scoreBonus = 10)
class CriticalPathTest { }

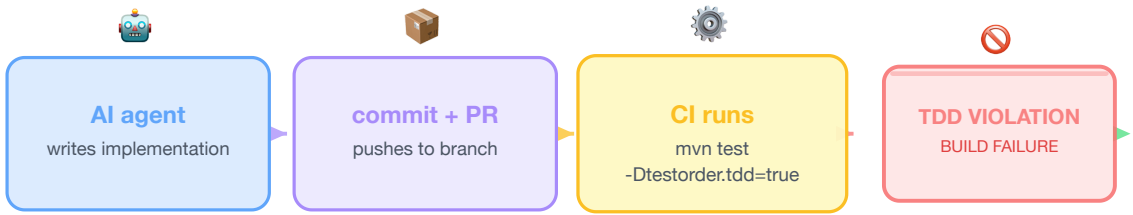
@TestOrder(changeBonus = 5)
class IntegrationTest { }
```

`testorder.tdd=true`

```
[test-order] TDD VIOLATION: NewFeatureTest
  → Test passed on first run with no prior failure recorded.
  → Write the failing test BEFORE the implementation.
BUILD FAILURE
```

> **Flip this on in CI. It will catch tests written after the feature, not before.**

AI agents + CI = TDD enforcement



The test-order skill — agents run it natively

Agent's inner loop

1. Make a code change
2. `mvn test-order:affected test`
3. Read the failure — fix root cause
4. Repeat until green
5. Open PR → CI runs full suite

Why it helps agents

- Affected tests only → **10× faster feedback**
- JSON output → easy to parse failures
- Skill file ships the exact commands
- No hallucinated flags — manifest gates every goal

`skills/test-order/SKILL.md` — drop it in your project, agents auto-load it

DEMO D6

Claude Code × test-order: fix a bug in < 60 s

```
cd third-party/spring-petclinic

# Agent already has skills/test-order/SKILL.md loaded.
# Introduce a bug, then hand off to the agent:

claude "There's a regression in VisitController. Find it,
        fix it, and verify with test-order:affected. Use
        the test-order skill."
```

The index never goes stale

Opcode hash — checked on every run. Bytecode changed → invalidate entry.

e.g. you edit `CartService.java` and recompile → its entry is removed, affected tests re-run uncached

Dep fingerprint — source change triggers selective re-learn for affected classes only.

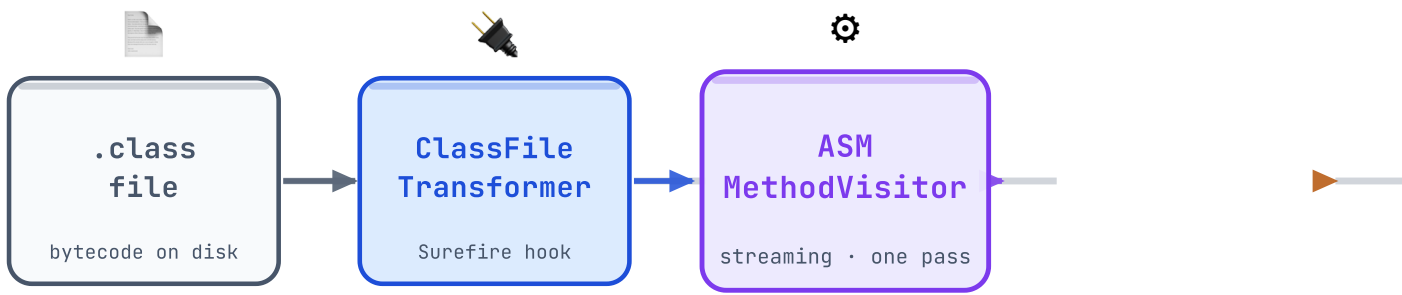
e.g. you add a new dependency to `pom.xml` → plugin switches to learn mode automatically

Unknown classes — over-approximated and deferred, never silently skipped.

e.g. a new class with no dep record → all tests that could possibly touch it are included

One `invokestatic` per method entry.

learn run only



Three modes. Similar overhead.

Mode	Tracks	Precision	Index size
`CLASS`	Which classes load	Coarsest	Smallest
`METHOD`	Per-method calls	Medium	Medium
`MEMBER` *(default)*	Field + method accesses	Most precise	Largest

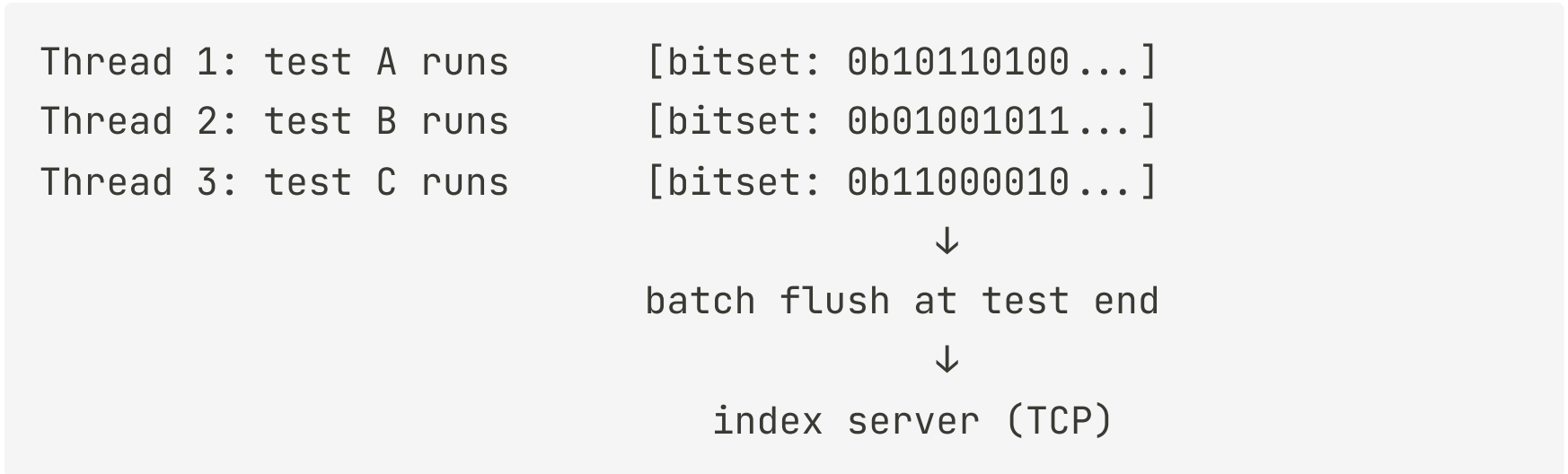
All three modes have similar overhead (~11–13%). The real differentiator is precision.
Use `CLASS` for huge legacy suites where index size matters. `MEMBER` works everywhere.

ASM · a handful of bytes per method

```
// Before instrumentation
public void remove() {
    last = null;
    iterator.remove();
}

// After
public void remove() {
    UsageStore.recordMemberUsageIdFast(42);
    last = null;
    iterator.remove();
}
```

Zero contention until test method ends



Questions?



`parttimenerd.github.io/test-order`

docs · repo



`feedback`

session rating



`sapmachine.io`

my team