

Speculative Script Reordering at Subprocess Granularity

Georgios Liargkovas*

Bolun Thompson

Diomidis Spinellis

Di Jin*

Anirudh Narsipur

Michael Greenberg

Ezri Zhu

Seong-Heon Jung

Konstantinos Kallas

Dan Liu

Siddhartha Prasad

Nikos Vasilakis



https://github.com/binpash/hs



* equal contribution

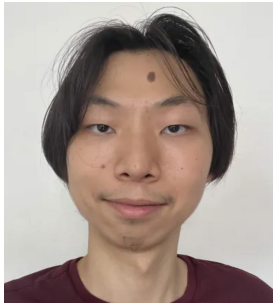
Joint work with…



Di Jin



Ezri Zhu



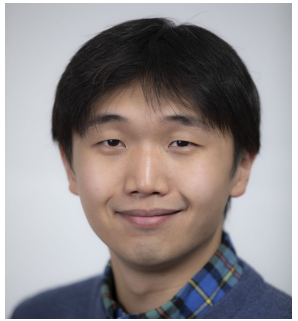
Dan Liu



Bolun Thompson



Anirudh Narsipur



Seong-Heon Jung



Siddartha Prasad



Diomidis Spinellis



Michael Greenberg



Konstantinos Kallas



Nikos Vasilakis

TL;DR



Prior shell optimization asks users to help (e.g., annotations, manual rewrites)

hS asks for nothing!

- **up to 6.2× speedup vs. Bash!**
- **beats prior work in many cases!**

Shell is ubiquitous

Builds + Deployment

- compilers
- test runners
- package managers
- deployment scripts
- log processing

Data + Systems

- ETL
- cloud CLIs
- filesystem scans
- web indexing
- ML preprocessing

Research Workflows

- RNA sequencing
- genome alignment
- post-processing (e.g., Bioinformatics).

The **fastest growing language** for AI repos **(+324% in 2025)**

Consistently in **top 10 languages**

Shell is ubiquitous

Builds + Deployment

- compiler
- test ru
- packa
- deploy
- log pr

Data + Systems

ETL

Research Workflows

RNA ing
nt
(e.g.,

Shell scripts are not only ubiquitous, but their performance matters for many workloads!

The **fastest growing language** for AI repos **(+324% in 2025)**

Consistently in **top 10 languages**

Shell is hard to optimize

Shell syntax

```
A | B

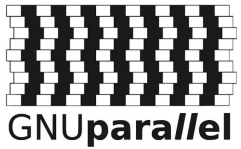
A & … ; wait
```

```
for x in ./*; do
     cmd "$x" &
done
wait
```

Command-specific flags

```
make -j
xargs -P
```

Generic tools




```
parallel cmd ::: ./*
```

Specialized systems



```
pash script.sh

posh script.sh
```

Shell is hard to optimize

Shell syntax

```
A | B
```

```
A & … ; wait
```

```
for x in ./*; do
    cmd "$x" &
done
```

All 4 require some form of user input to get their benefits.
Not always easy, feasible, or possible!

Specialized systems



```
pash script.sh
```

```
posh script.sh
```

A performance-critical shell script

```
./prep_db.sh db/sqlite.db                    # a

for SM in $SAMPLES; do

  samtools index $SM/Genome.sorted.bam       # b

  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome             # c
done
```

[TERA-Seq] Ibrahim, F., Oppelt, J., Maragkakis, M., and Mourelatos, Z. 2021. TERA-Seq: true end-to-end sequencing of native RNA molecules for transcriptome characterization. *Nucleic Acids Res.* 49, 20, e115.

A performance-critical shell script

Prepare DB

```
./prep_db.sh db/sqlite.db                    # a

for SM in $SAMPLES; do

  samtools index $SM/Genome.sorted.bam       # b

  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome             # c
done
```

[TERA-Seq] Ibrahim, F., Oppelt, J., Maragkakis, M., and Mourelatos, Z. 2021. TERA-Seq: true end-to-end sequencing of native RNA molecules for transcriptome characterization. *Nucleic Acids Res.* 49, 20, e115.

A performance-critical shell script

Prepare DB

Iterate over samples

```
./prep_db.sh db/sqlite.db                    # a

for SM in $SAMPLES; do

  samtools index $SM/Genome.sorted.bam       # b

  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome             # c
done
```

[TERA-Seq] Ibrahim, F., Oppelt, J., Maragkakis, M., and Mourelatos, Z. 2021. TERA-Seq: true end-to-end sequencing of native RNA molecules for transcriptome characterization. *Nucleic Acids Res.* 49, 20, e115.

A performance-critical shell script

Prepare DB

Iterate over samples

Build index

```
./prep_db.sh db/sqlite.db                    # a

for SM in $SAMPLES; do

  samtools index $SM/Genome.sorted.bam       # b

  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome             # c
done
```

[TERA-Seq] Ibrahim, F., Oppelt, J., Maragkakis, M., and Mourelatos, Z. 2021. TERA-Seq: true end-to-end sequencing of native RNA molecules for transcriptome characterization. *Nucleic Acids Res.* 49, 20, e115.

A performance-critical shell script

Prepare DB

Iterate over samples

Build index

Query sample & save to DB

```
./prep_db.sh db/sqlite.db                    # a

for SM in $SAMPLES; do

  samtools index $SM/Genome.sorted.bam       # b

  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome             # c
done
```

[TERA-Seq] Ibrahim, F., Oppelt, J., Maragkakis, M., and Mourelatos, Z. 2021. TERA-Seq: true end-to-end sequencing of native RNA molecules for transcriptome characterization. *Nucleic Acids Res.* 49, 20, e115.

A performance-critical shell script

Prepare DB

Iterate over samples

Build index

Query sample & save to DB

```
./prep_db.sh db/sqlite.db                    # a

for SM in $SAMPLES; do

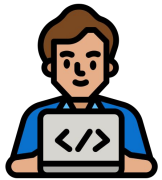
  samtools index $SM/Genome.sorted.bam       # b

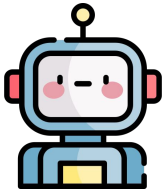
  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome             # c
done
```

17 minutes

[TERA-Seq] Ibrahim, F., Oppelt, J., Maragkakis, M., and Mourelatos, Z. 2021. TERA-Seq: true end-to-end sequencing of native RNA molecules for transcriptome characterization. *Nucleic Acids Res.* 49, 20, e115.

Rewrite manually or with an AI agent?





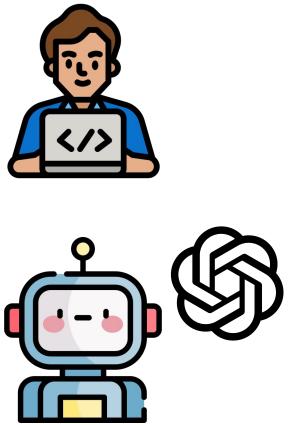
```
./prep_db.sh db/sqlite.db                          # a

for SM in $SAMPLES; do

  samtools index $SM/Genome.sorted.bam             # b

  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome                   # c
done
```

Rewrite manually or with an AI agent?



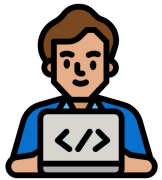
```
./prep_db.sh db/sqlite.db                          # a

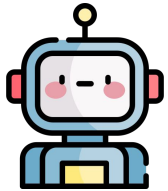
for SM in $SAMPLES; do

  samtools index $SM/Genome.sorted.bam &    # b

  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome       &    # c
done
wait
```

Problem 1: Syntax doesn't tell us everything





Dependency

Run c_i only after b_i

Hidden index file



writes ←

reads →

```
./prep_db.sh db/sqlite.db                          # a

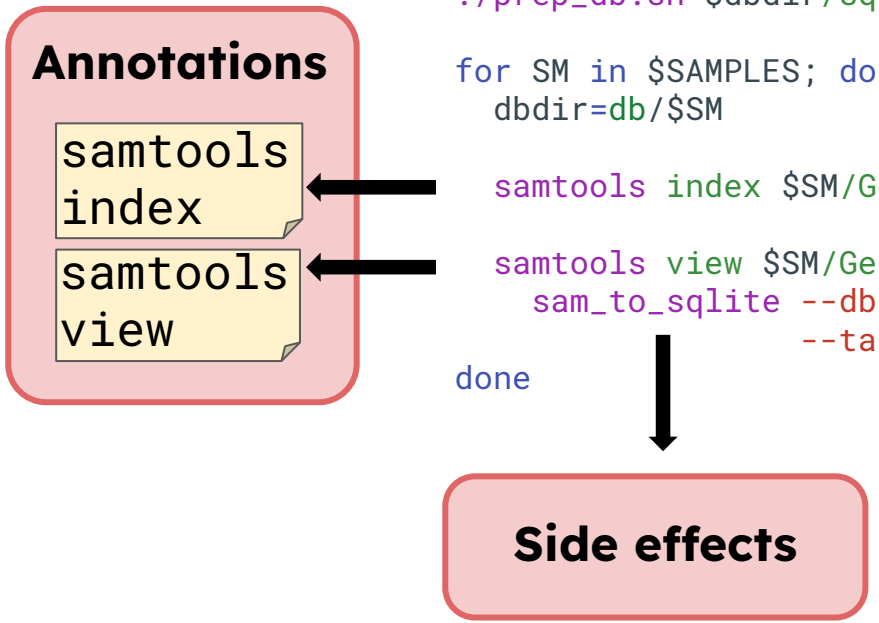
for SM in $SAMPLES; do

    samtools index $SM/Genome.sorted.bam &         # b

    samtools view $SM/Genome.sorted.bam $REGION |
      sam_to_sqlite --db db/sqlite.db \
                    --table genome       &         # c
done
wait
```

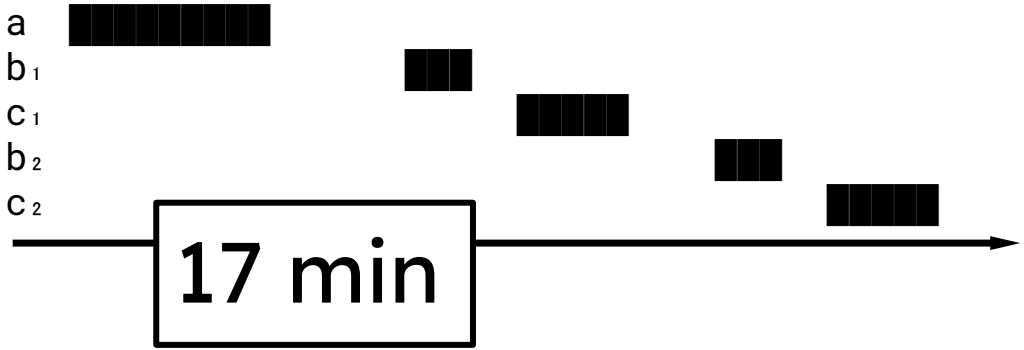
Problem 2: PaSh needs annotations





Challenge 1: Script order is not always optimal

Bash execution order




```
./prep_db.sh db/sqlite.db                    # a

for SM in $SAMPLES; do

  samtools index $SM/Genome.sorted.bam       # b

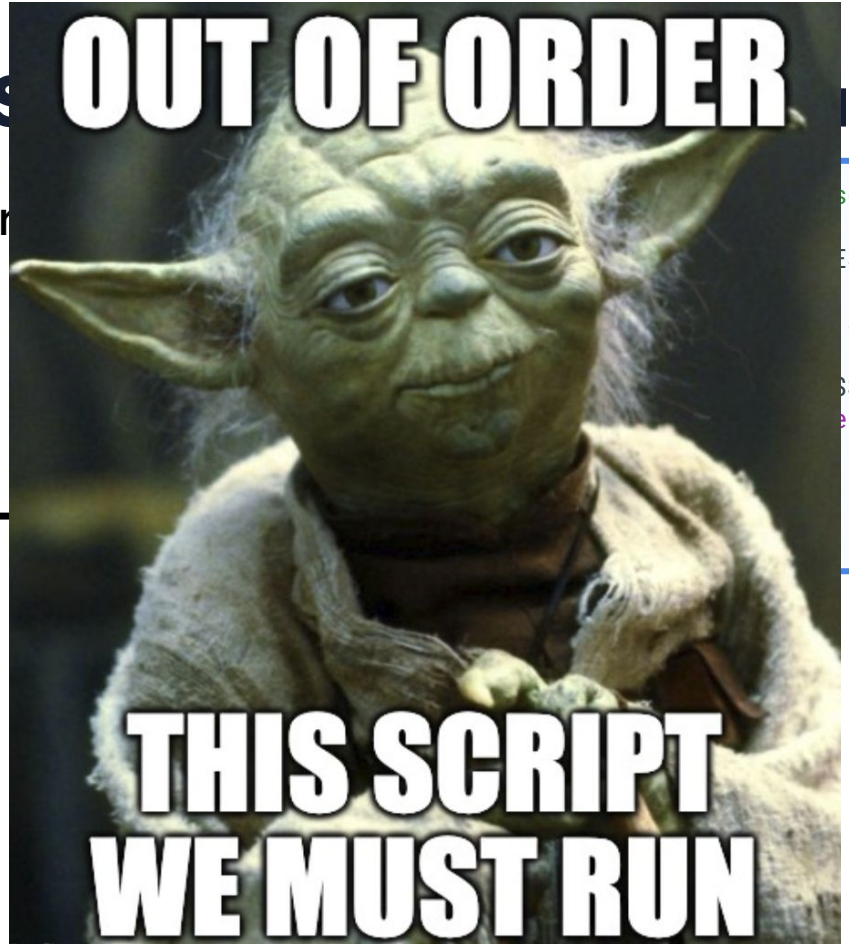
  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome             # c
done
```

Challenge 1: S ys optimal

Bash execution order

a
b₁
c₁
b₂
c₂

17 min




```
sqlite.db                          # a

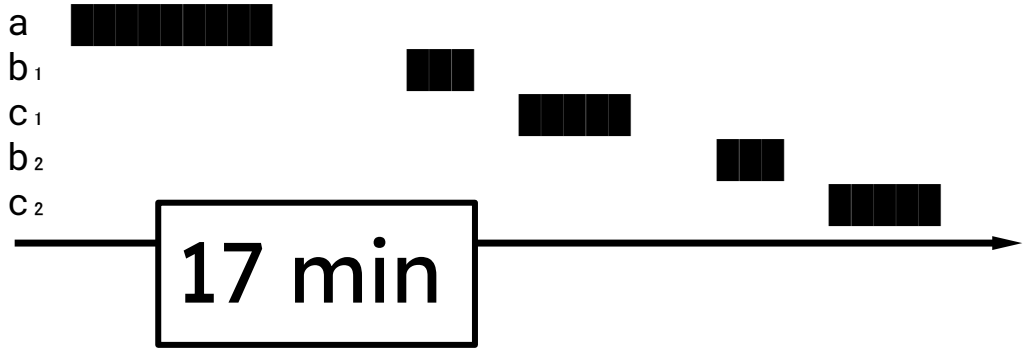
ES; do

 $SM/Genome.sorted.bam             # b

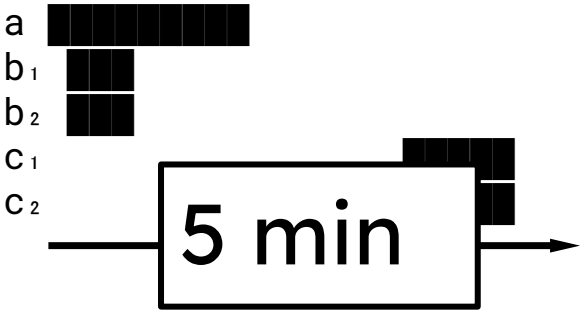
$SM/Genome.sorted.bam $REGION |
e --db db/sqlite.db \
  --table genome                   # c
```

Solution: Out-of-order Execution

Bash execution order



Optimal order




```
./prep_db.sh db/sqlite.db                          # a

for SM in $SAMPLES; do

  samtools index $SM/Genome.sorted.bam             # b

  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome                   # c
done
```

Dependencies

- Run c_i only after **a**
- Run c_i only after b_i

Run **a** and b_i out-of-order



Speculation: Run commands early, out-of-order.



Challenge 2: Keep effects private.

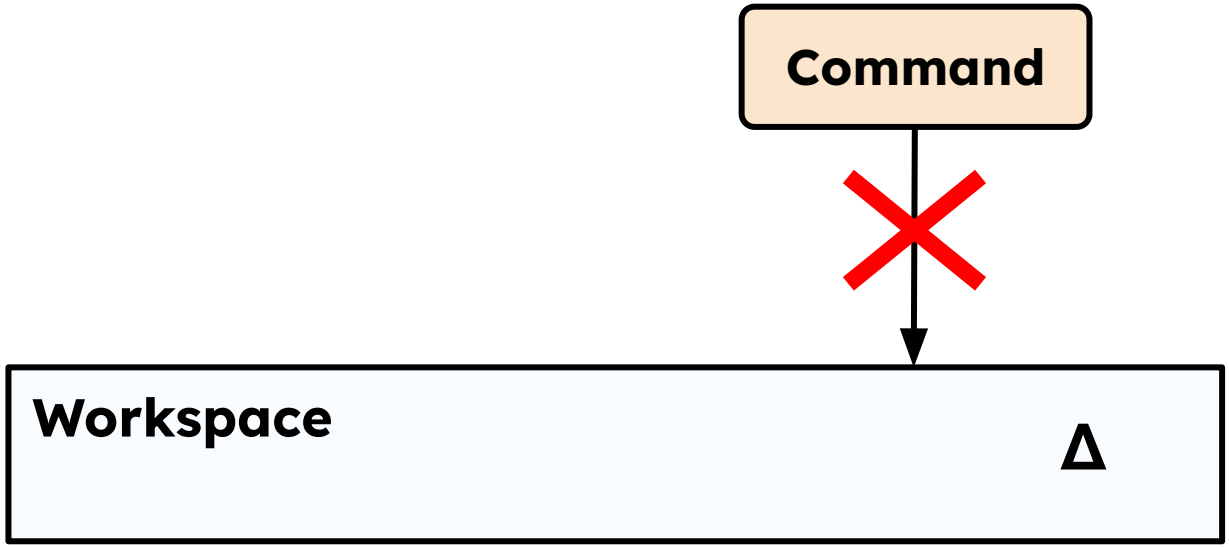


Challenge 3: Discover runtime dependencies.

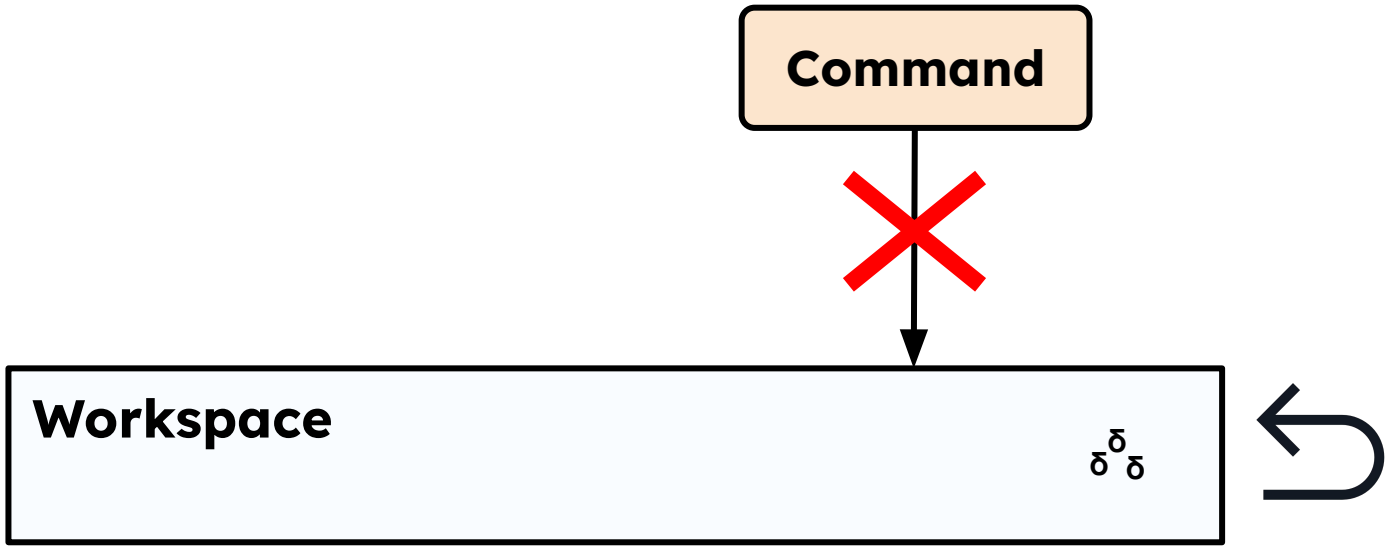




Challenge 2: Command effect isolation

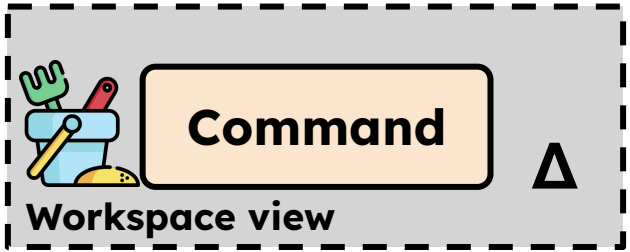


Challenge 2: Command effect isolation





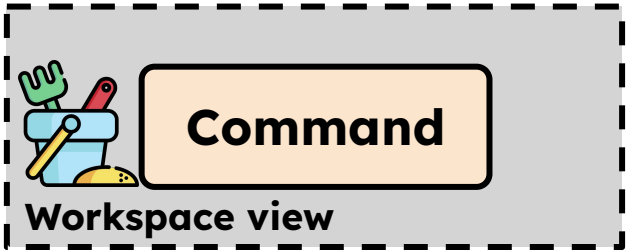
Sandboxing for effect isolation



Workspace



Sandboxing for effect isolation

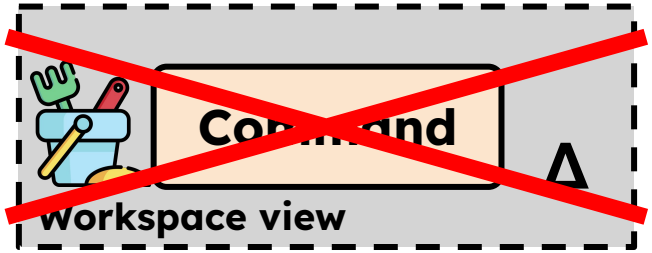








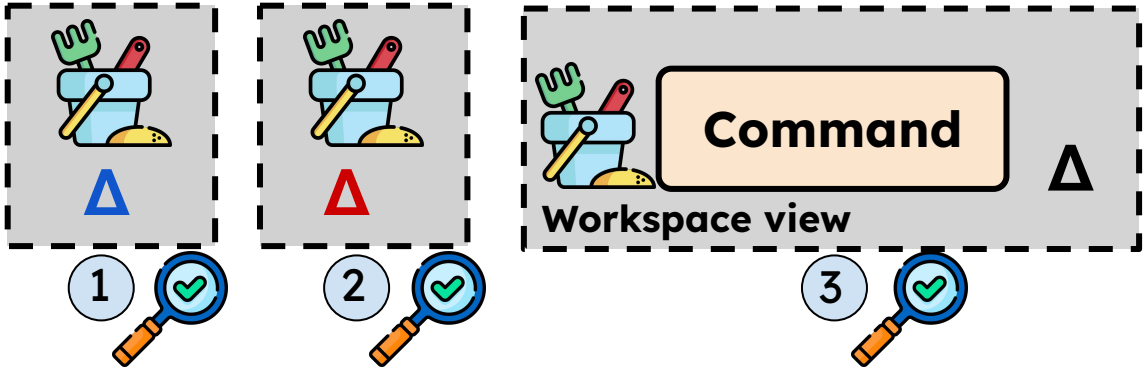
Sandboxing for effect isolation

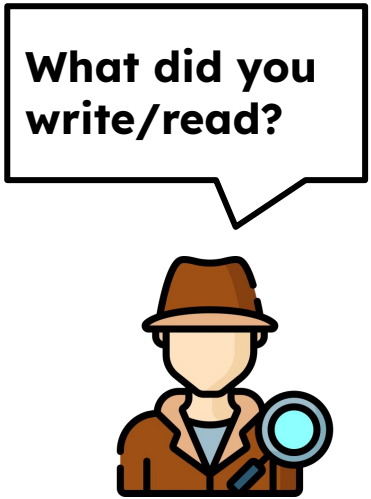


Workspace



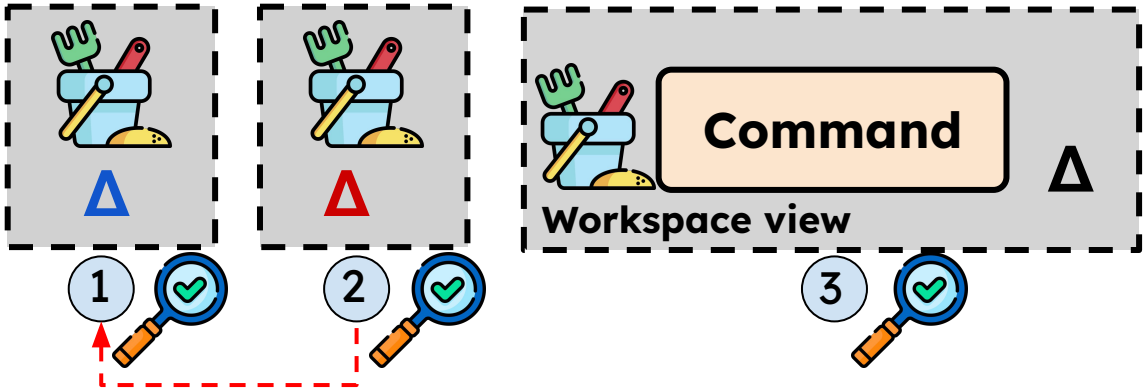
Tracing for dependency resolution





Workspace

Tracing for dependency resolution



What did you write/read?



Workspace

Dependencies

Run **2** only after **1**

hS Contributions

Speculation: Run commands early, out-of-order.



Semisolates: Keep effects private.



Tracing: Discover runtime dependencies.



hS Contributions

Speculation: Run commands early, out-of-order.



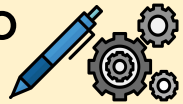
Semisolates: Keep effects private.



Tracing: Discover runtime dependencies.



Preprocess & Runtime: Decide what and when to rerun & commit in-order.





Formally Verified Runtime State



hS Runtime

Orchestrator

Schedule commands

Executor

Run commands in isolation



hS Runtime

Orchestrator

Schedule

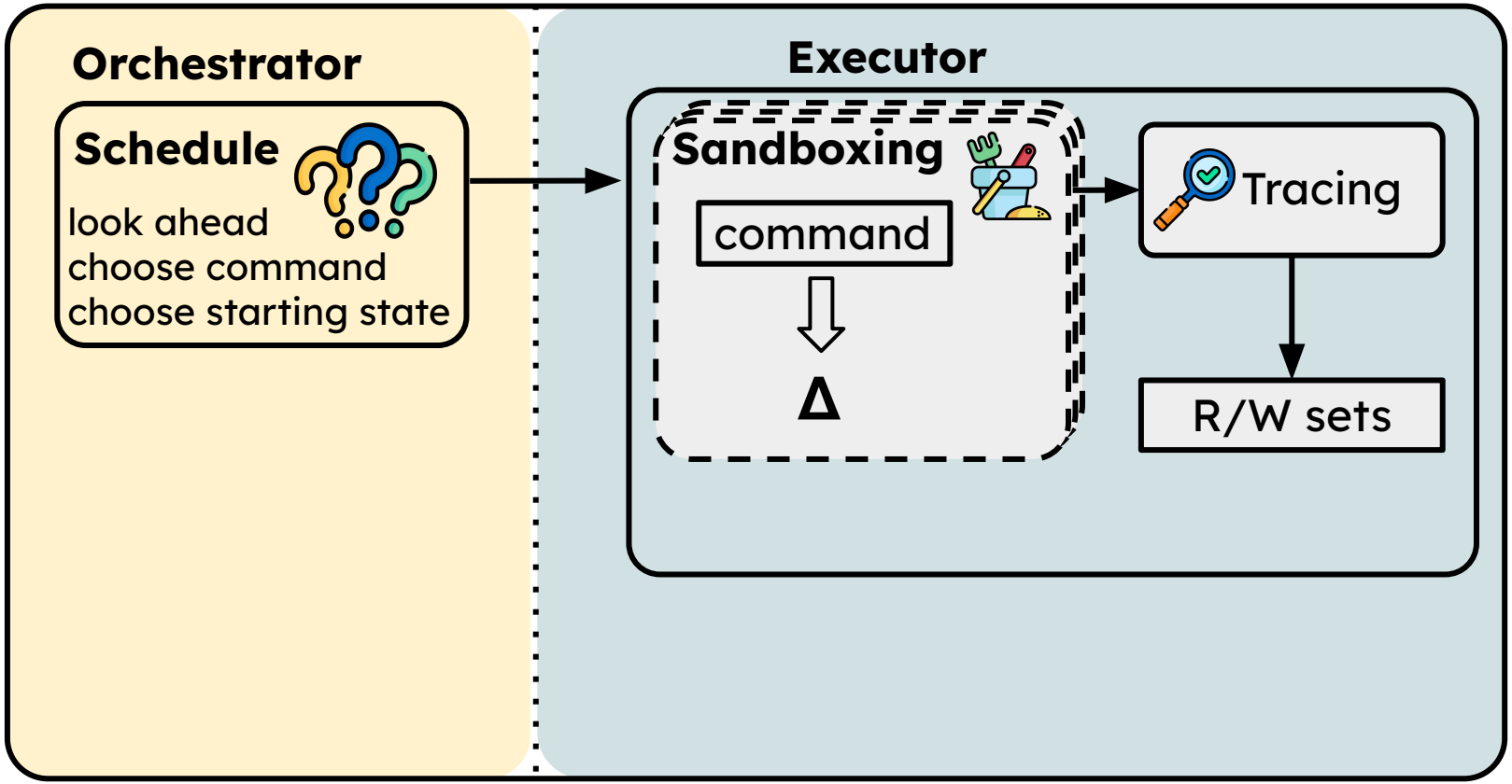


look ahead
choose command
choose starting state

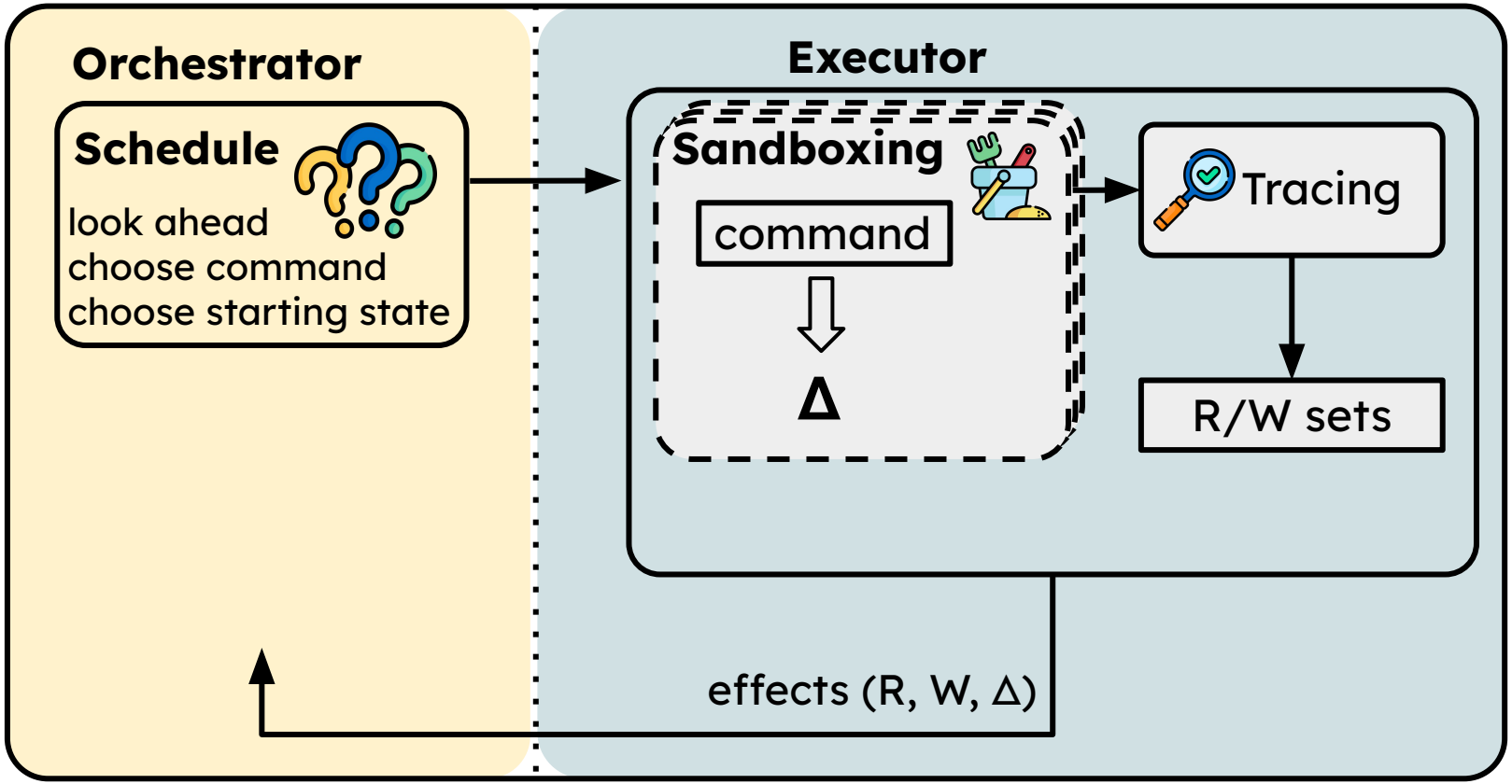
Executor

Run commands
in isolation

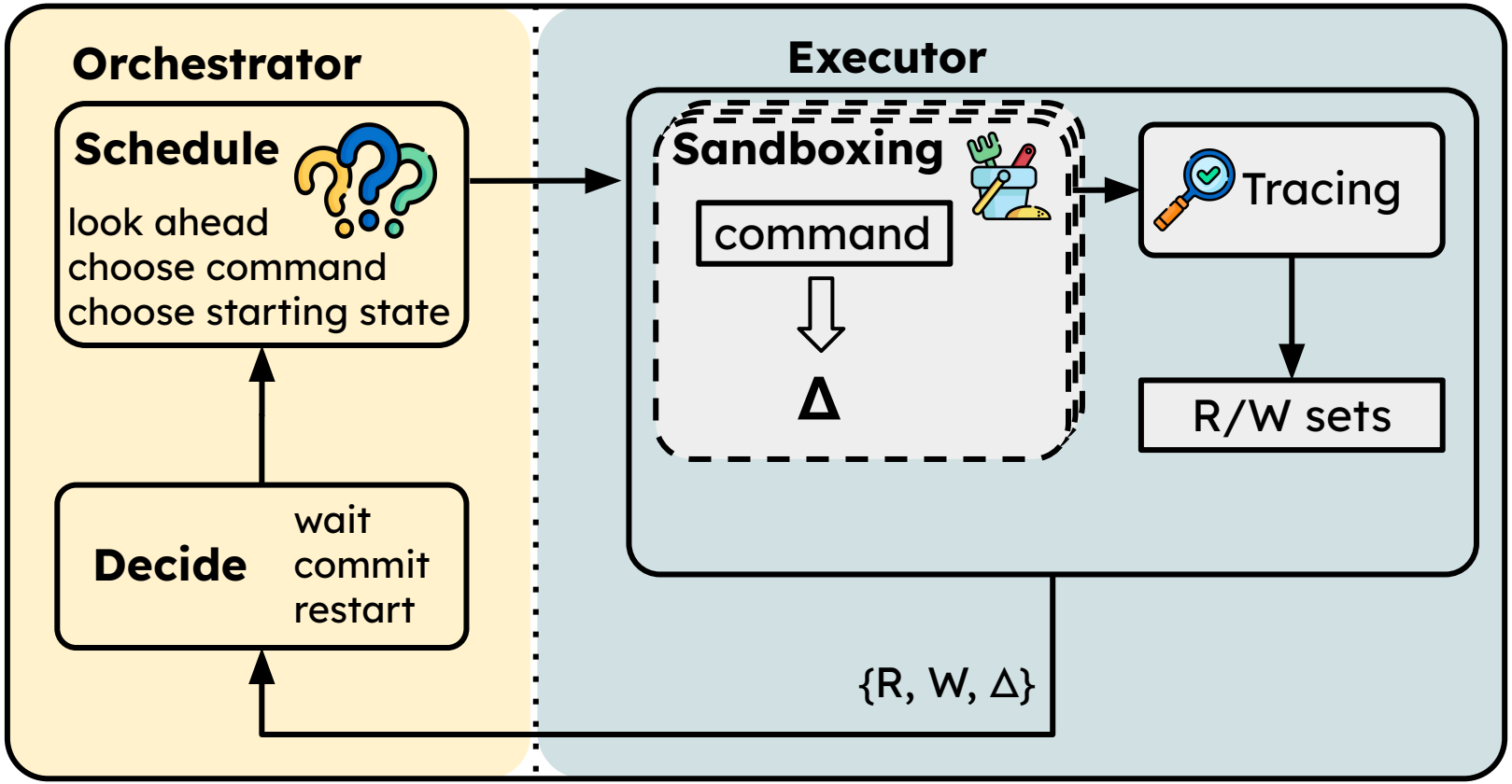
hS Runtime


Orchestrator
Schedule
look ahead
choose command
choose starting state
Executor
Sandboxing
command
Δ
Tracing
R/W sets

hS Runtime


Orchestrator
Schedule
look ahead
choose command
choose starting state
Executor
Sandboxing
command
Δ
Tracing
R/W sets
effects (R, W, Δ)

hS Runtime


Orchestrator
Schedule
look ahead
choose command
choose starting state
Decide
wait
commit
restart
Executor
Sandboxing
command
Δ
Tracing
R/W sets
{R, W, Δ}



The in-order illusion

Original script

```
./prep_db.sh db/sqlite.db                    # a

for SM in $SAMPLES; do

  samtools index $SM/Genome.sorted.bam       # b

  samtools view $SM/Genome.sorted.bam $REGION |
    sam_to_sqlite --db db/sqlite.db \
                  --table genome             # c
done
```

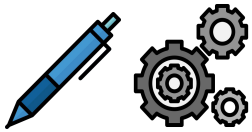
Preprocessor



Program skeleton

```
hs_request(a)

for SM in $SAMPLES;
do
  hs_request(b)
  hs_request(c)
done
```

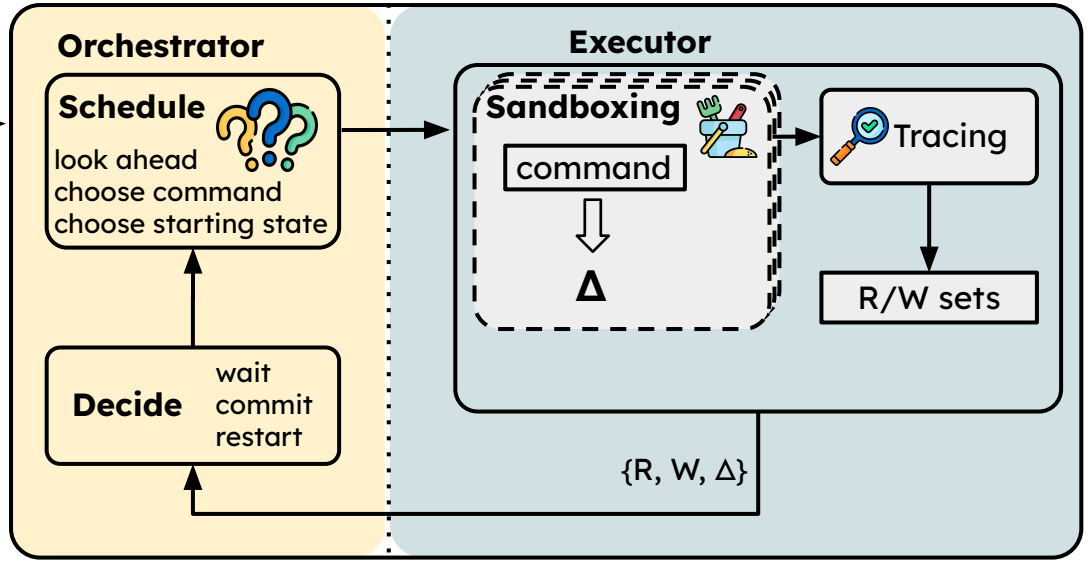


Keeping things in-order

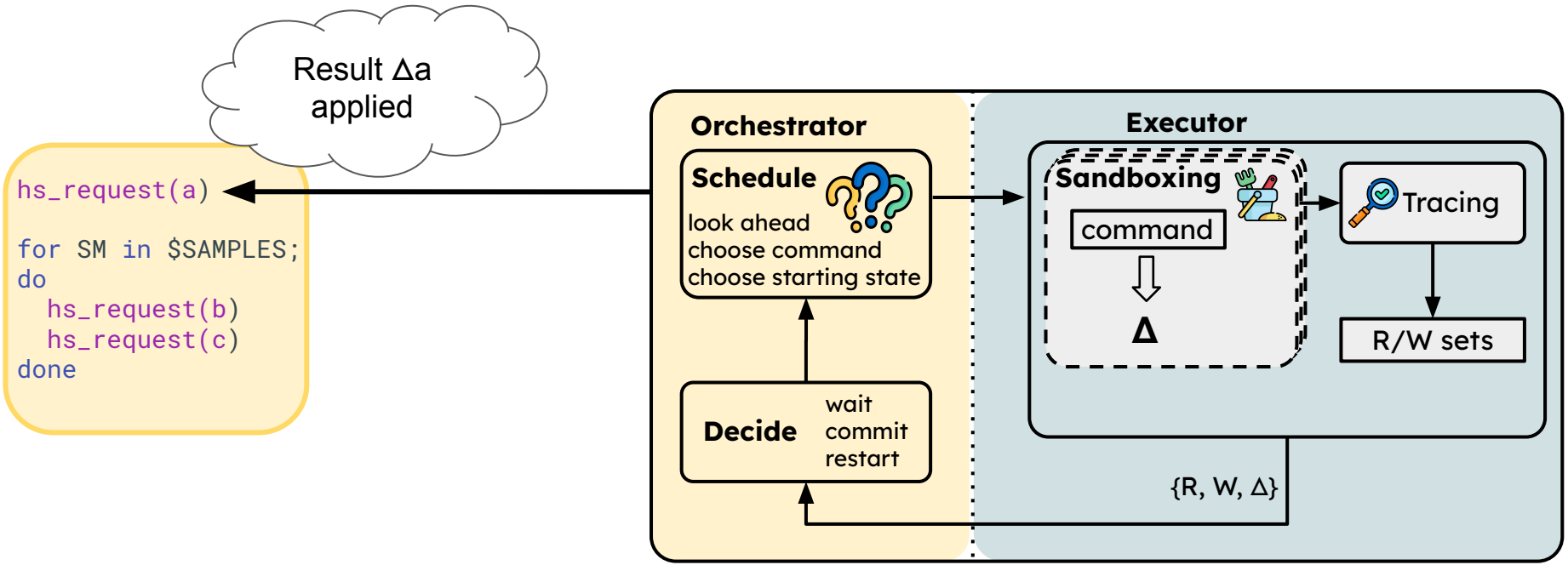
Blocking request:
"give me result of a"

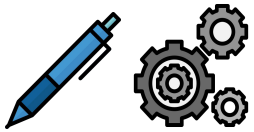
```
hs_request(a)

for SM in $SAMPLES;
do
  hs_request(b)
  hs_request(c)
done
```



Keeping things in-order



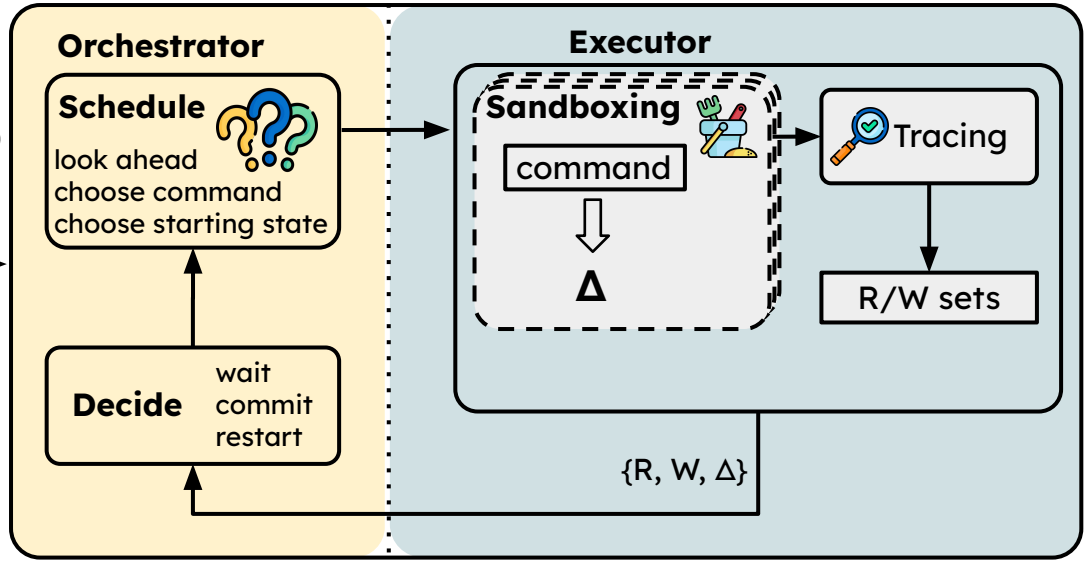


Keeping things in-order

```
hs_requ

for SM i
do
  hs_request(b)
  hs_request(c)
done
```

Blocking request:
"give me result of b"



More in the paper!

Implementation Details

- **Executor** details
- **Tracer** implementation
- **Sandboxing** with Try
- Orchestrator **state machine**
- **Control flow** handling

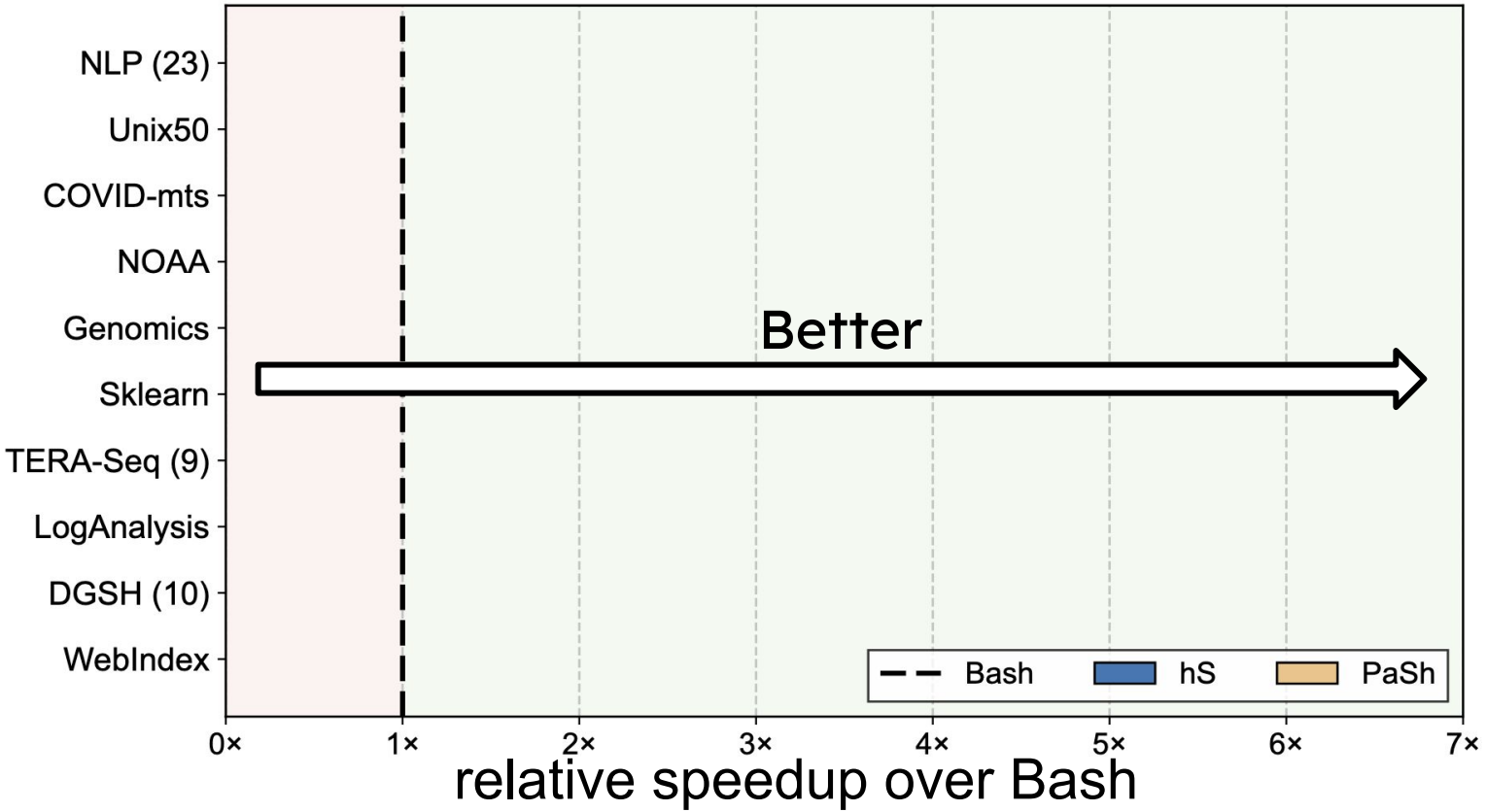


Optimizations

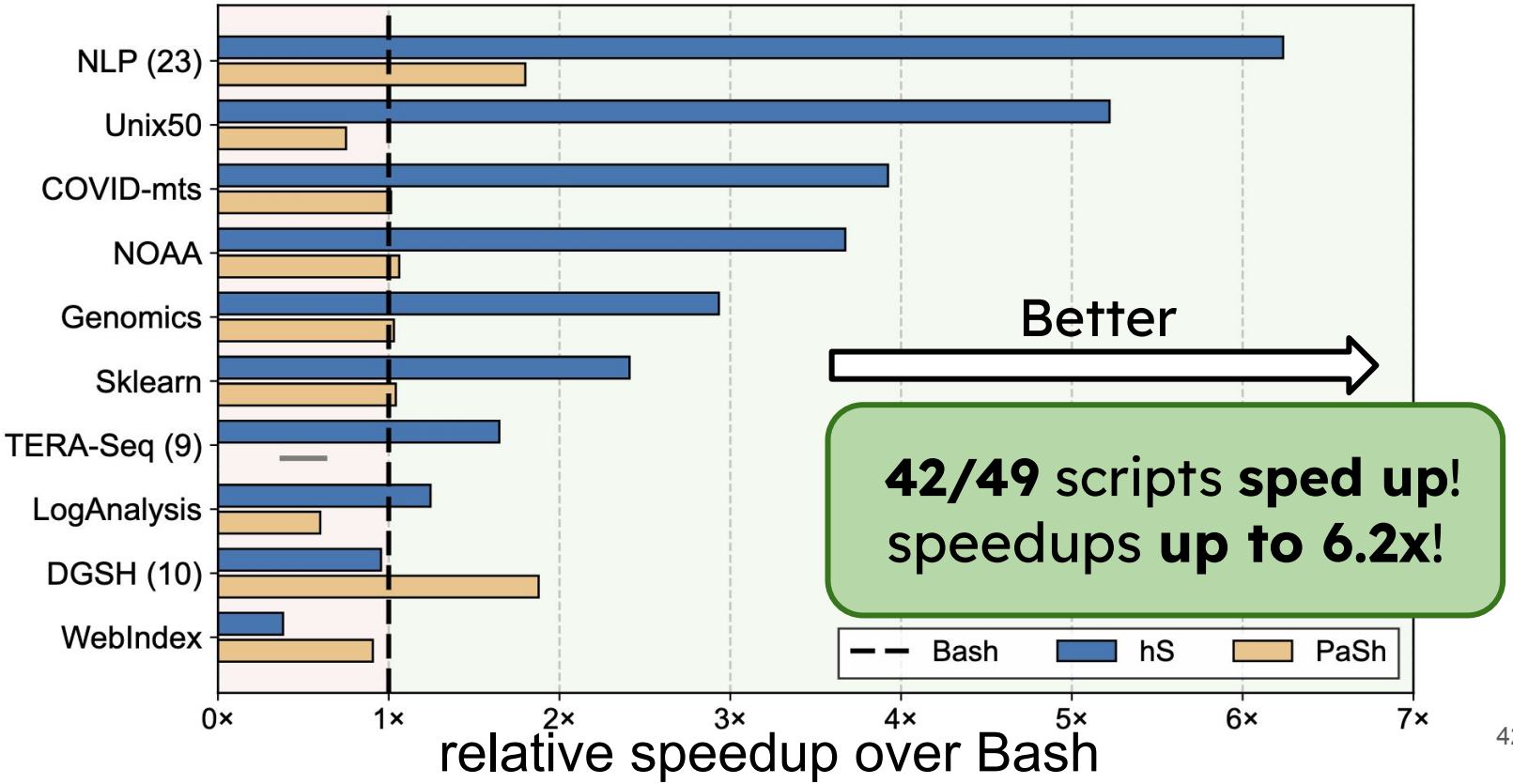
- Proactive **conflict detection**
- **Loop unrolling**
- **Speculation of speculated commands**



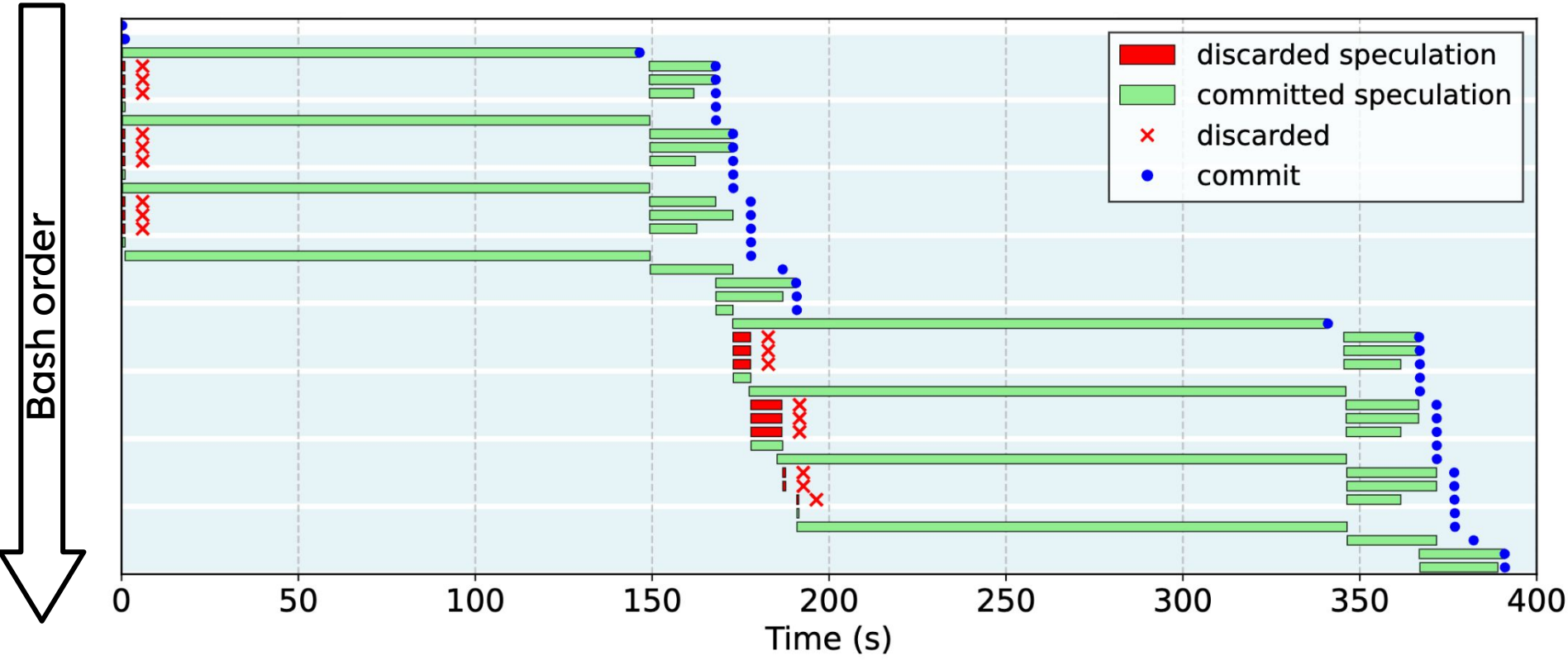
Speedup over Bash



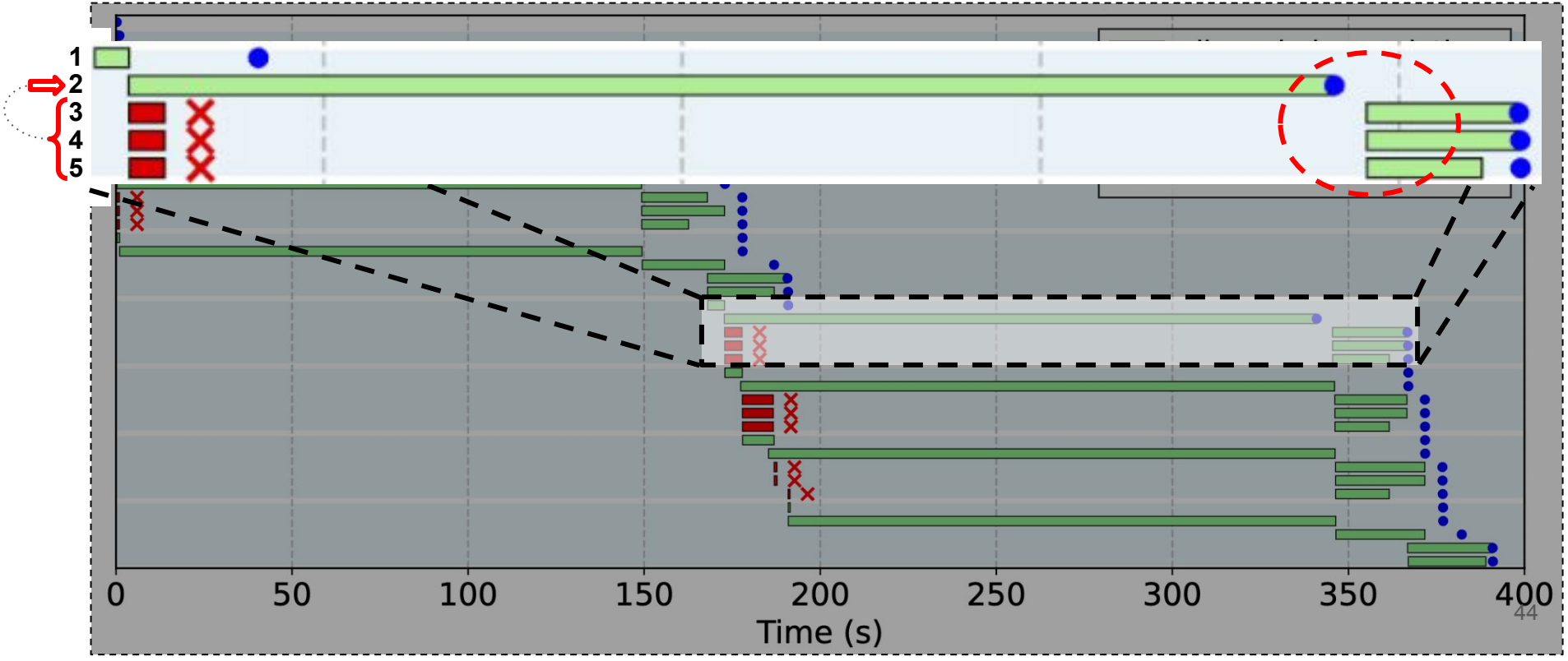
Speedup over Bash



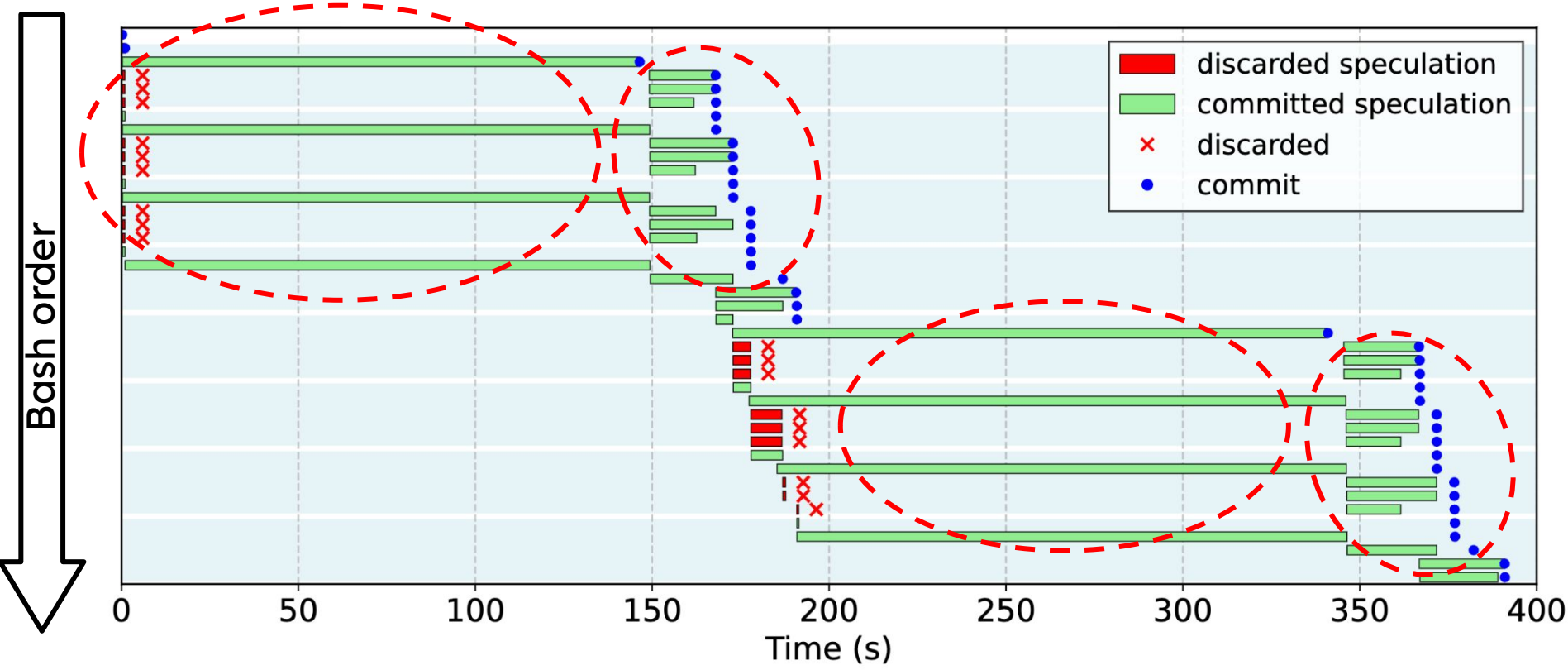
Speculation zoom-in



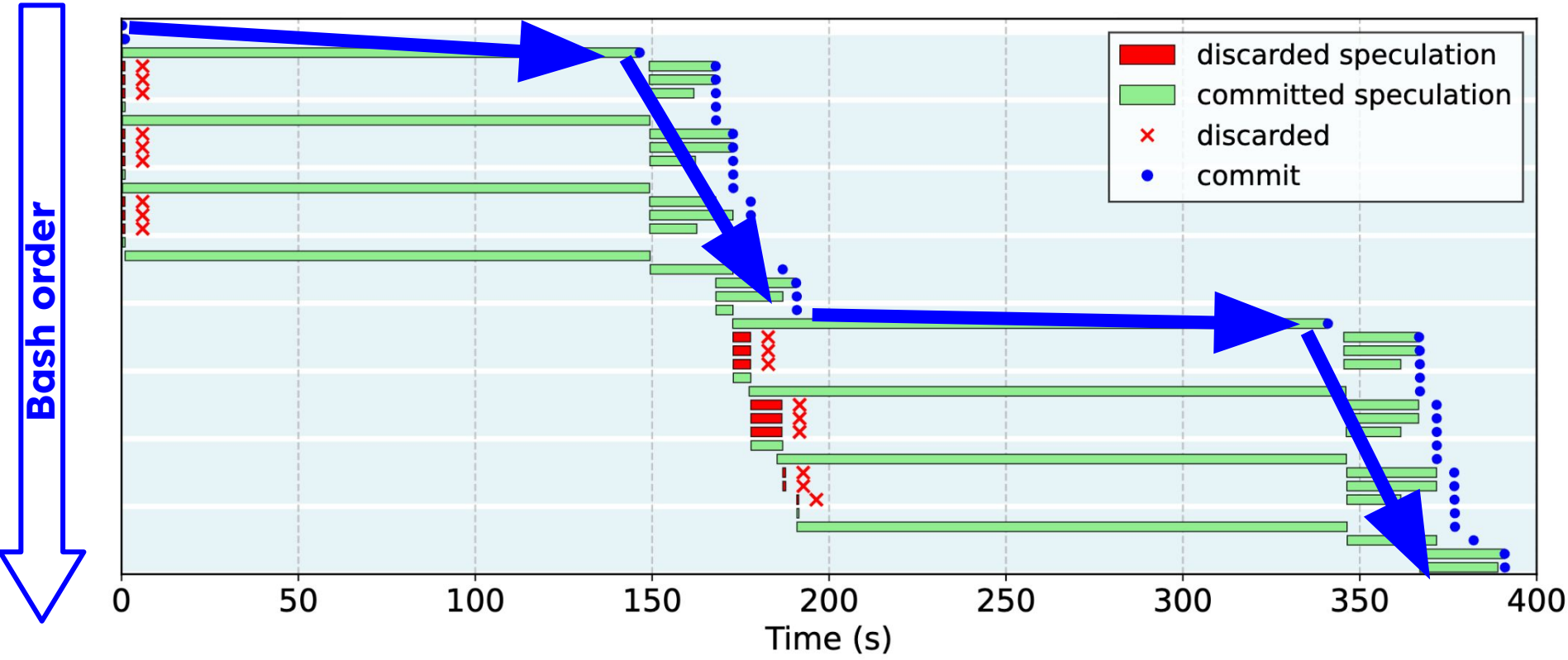
Speculation zoom-in



Speculation zoom-in



Speculation zoom-in



hS enables out-of-order shell execution

- Shell scripts schedule critical work, but expose **one** order
- hS optimizes shell scripts **without any user input**
- **out-of-order + sandbox + trace + orchestrate** → commit **in-order**
- **42 / 49 scripts** speed up (up to 6.2×)

Thank you!

- hS is open source!

```
$ hs my_script.sh
```

- **More in the paper:**
  - Executor
  - Tracer
  - Sandboxing with try
  - Orchestrator model checking
  - Optimizations
  - Limitations

Try it on the script you were afraid to rewrite!

ARTIFACT EVALUATED
usenix ASSOCIATION
AVAILABLE

Paper

GitHub Repo

Georgios Liargkovas
gl2902@columbia.edu

https://github.com/binpash/hs