Is the Problem-Based Benchmark Suite Fearless with Rust?

Javad Abdi, Guowei Zhang, Mark C. Jeffrey

SPAA 2023

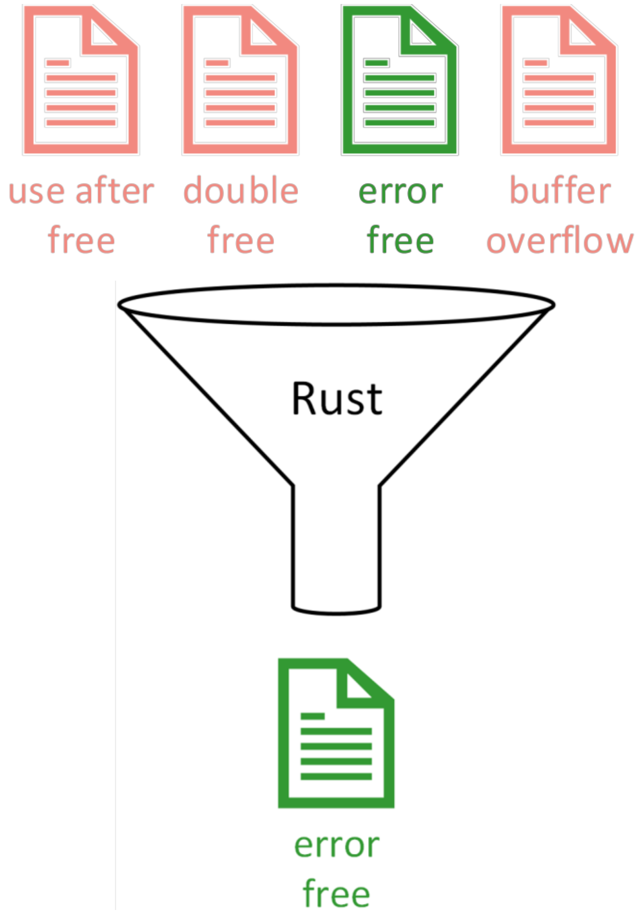
UNIVERSITY OF TORONTO

Rust is gaining popularity because of its safety guarantees

Rust is gaining popularity because of its safety guarantees

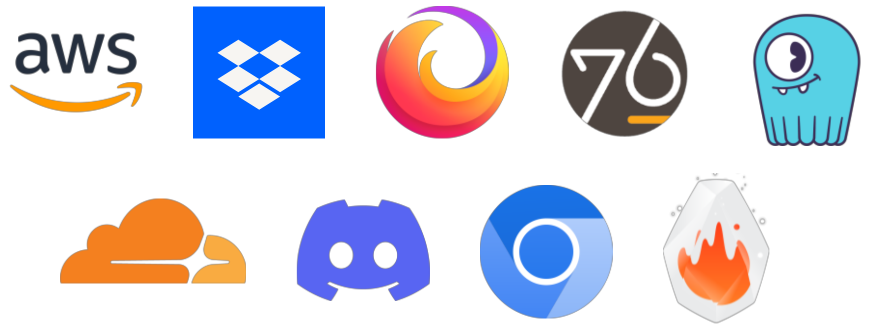
2022 Developer Survey

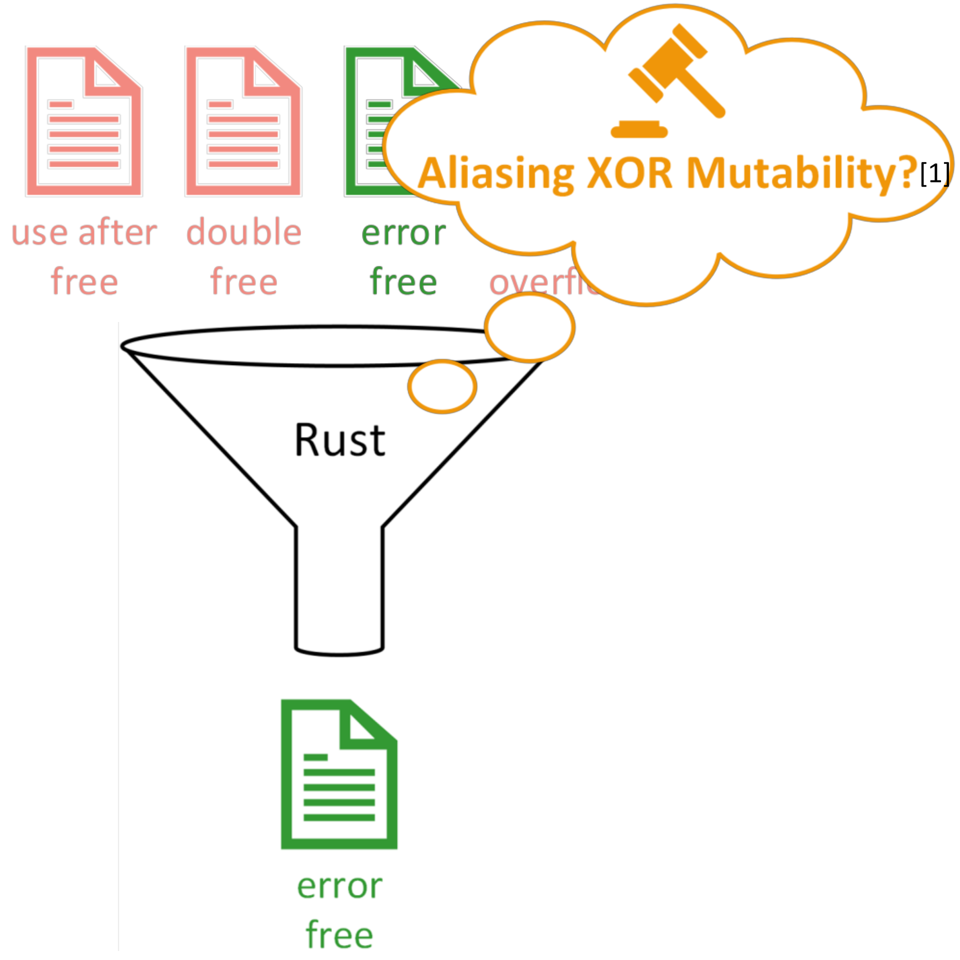
Rust is on its seventh year as the most loved language …



Rust is gaining popularity because of its safety guarantees

Rust is on its seventh year as the most loved language …



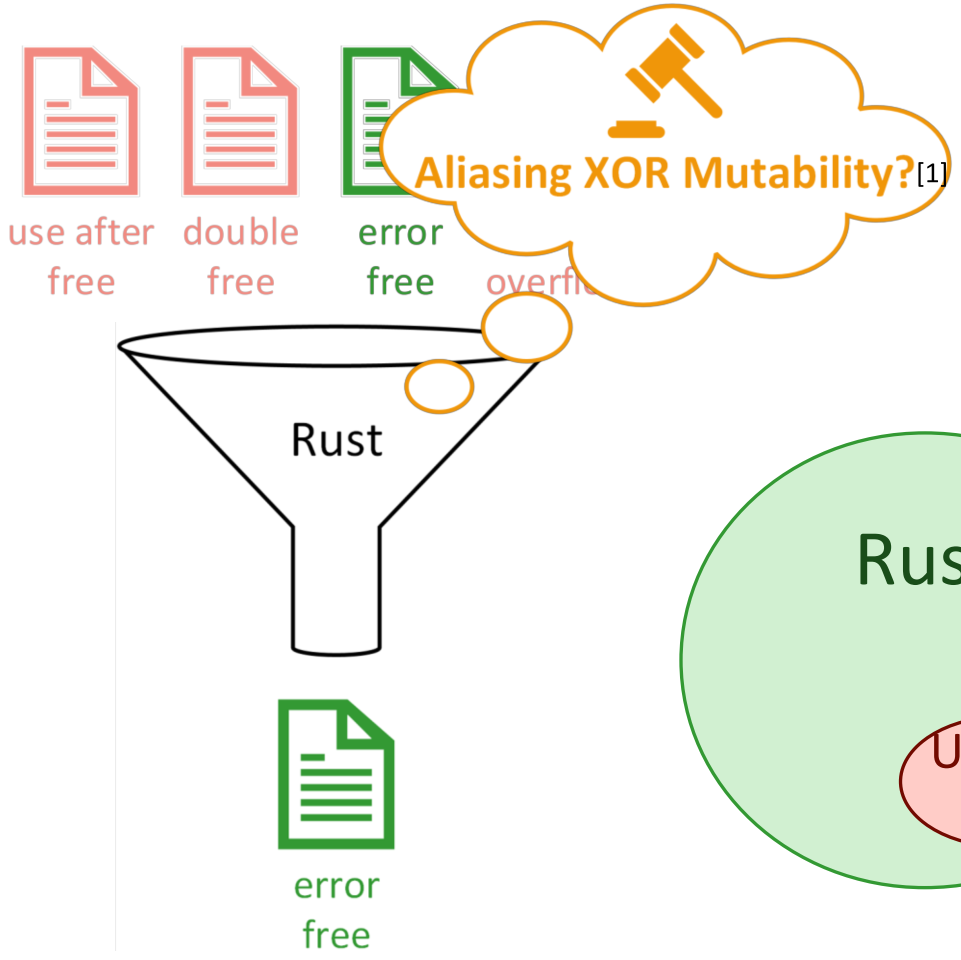


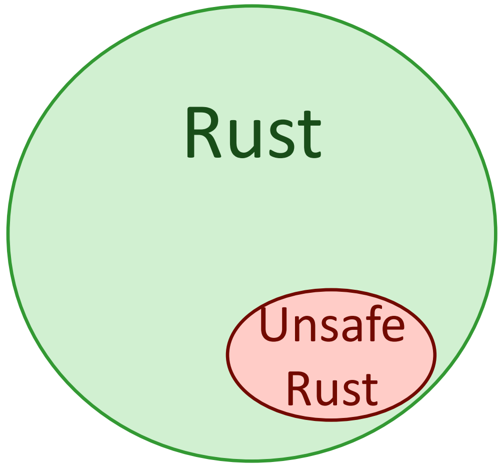
[1] Yanovski et al., ICFP 2021, GhostCell: separating permissions from data in Rust.

Rust is gaining popularity because of its safety guarantees

2022 Developer Survey

Rust is on its seventh year as the most loved language ...



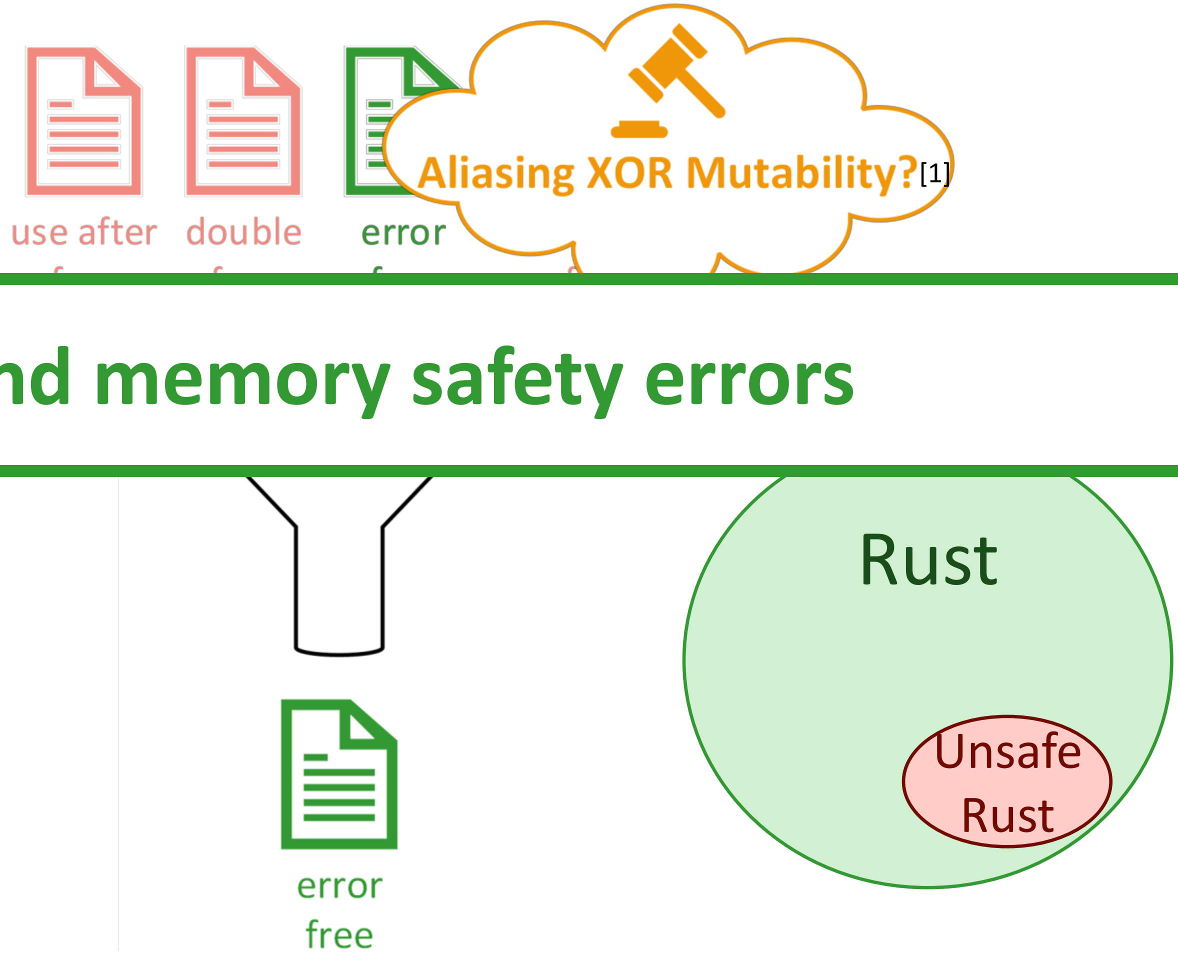


[1] Yanovski et al., ICFP 2021, GhostCell: separating permissions from data in Rust.

Rust is gaining popularity because of its safety guarantees

2022 Developer Survey

Rust is on its seventh year as the most loved language



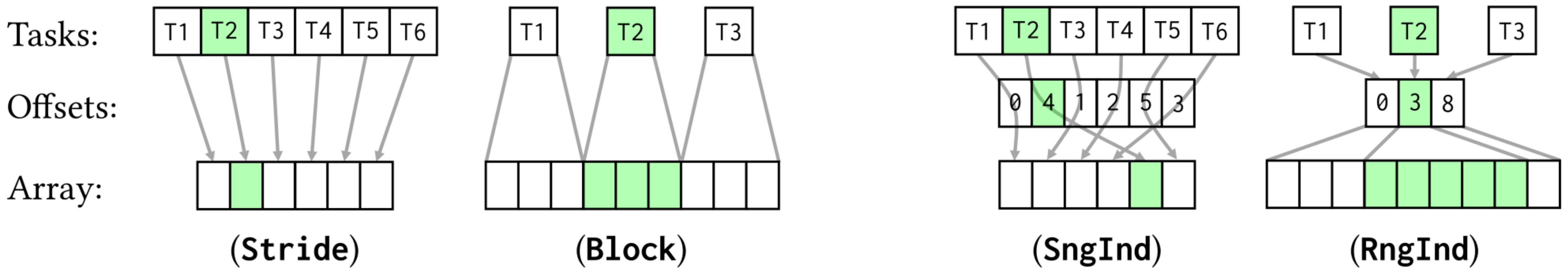
Rust catches all type and memory safety errors

[1] Yanovski et al., ICFP 2021, GhostCell: separating permissions from data in Rust.

Rust claims to provide "fearless concurrency"

Fear : Anticipation of concurrency errors that manifest at run time.

Our RQ : How does fearless concurrency translate to parallelism?

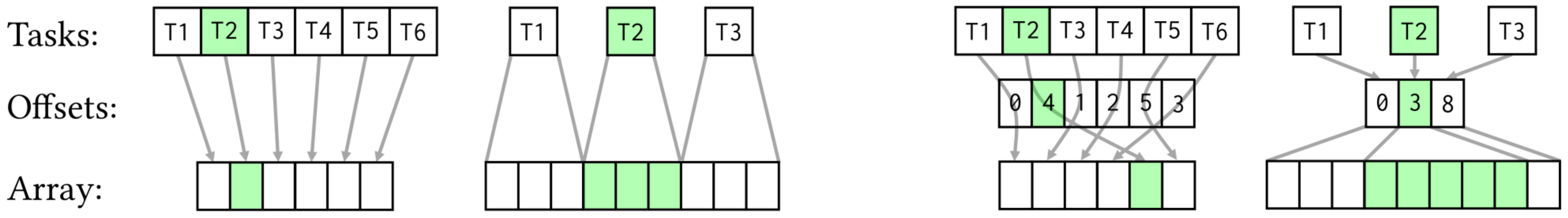


Rust claims to provide "fearless concurrency"

Fear : Anticipation of concurrency errors that manifest at run time.

Our RQ : How does fearless concurrency translate to parallelism?

Are all parallel patterns fearless in Rust?



(**Stride**) (**Block**) (**SngInd**) (**RngInd**)

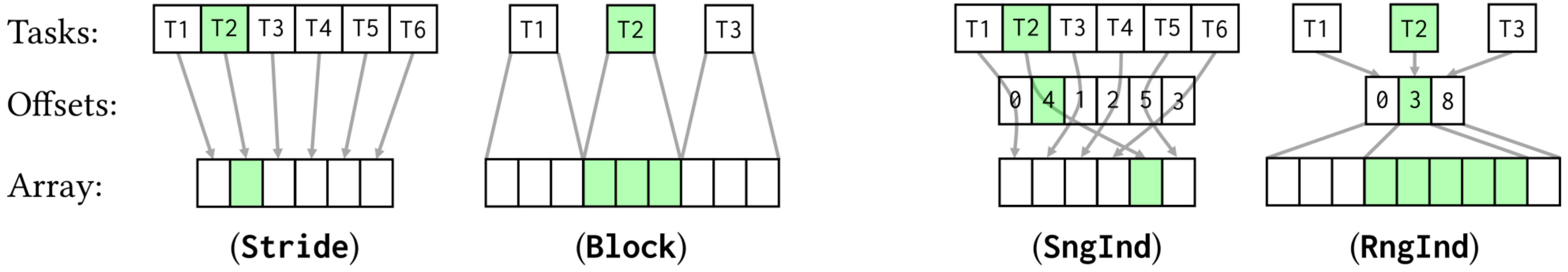
Contribution: Interrogate fearless concurrency by expressing (ir)regular parallelism

Rusty-PBBS:

- A port of PBBS[Anderson et al.,PPoPP'22] in Rust with both regular and irregular patterns.

Our Case Study:

- Classification of parallel expression patterns in Rusty-PBBS.
- Evaluating Rust support and fearlessness for each pattern.



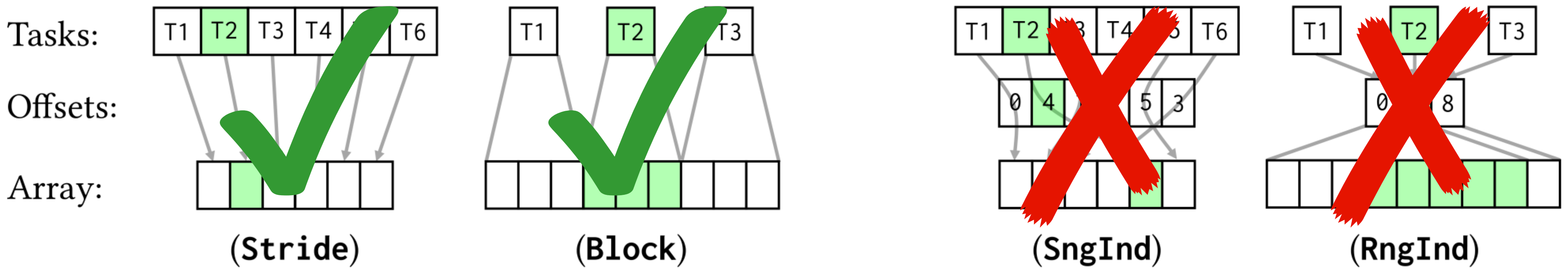
Contribution: Interrogate fearless concurrency by expressing (ir)regular parallelism

Rusty-PBBS:

- A port of PBBS[Anderson et al.,PPoPP'22] in Rust with both regular and irregular patterns.

Our Case Study:

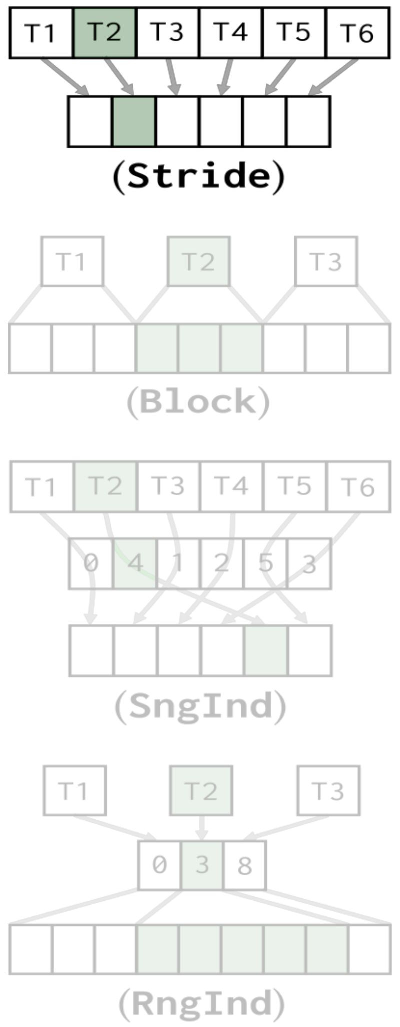
- Classification of parallel expression patterns in Rusty-PBBS.
- Evaluating Rust support and fearlessness for each pattern.



Fearless regular parallelism with Rust(+Rayon)

Regular parallelism:

- Known set of tasks
- Known dependences

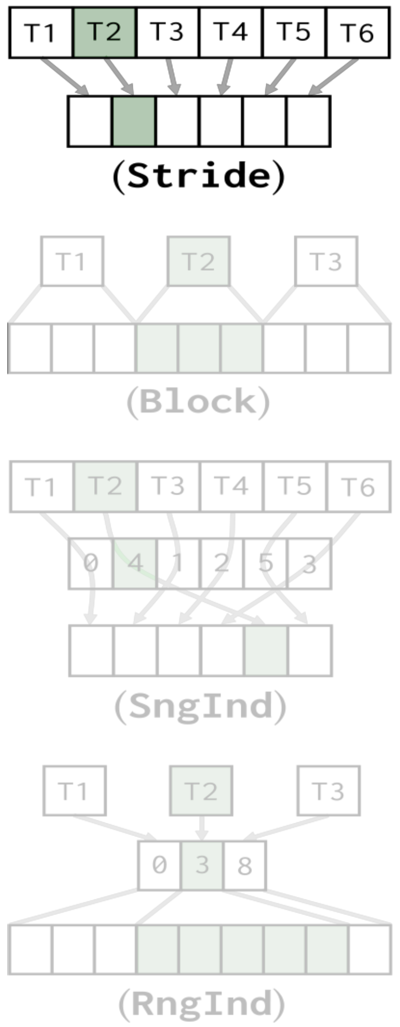


Fearless regular parallelism with Rust(+Rayon)

Regular parallelism:
- Known set of tasks
- Known dependences

```
fn par_increment(v: &mut [u32])
{
  v.par_iter_mut()
   .for_each(|vi|  *vi+=1);
}
```

stride pattern on **v**



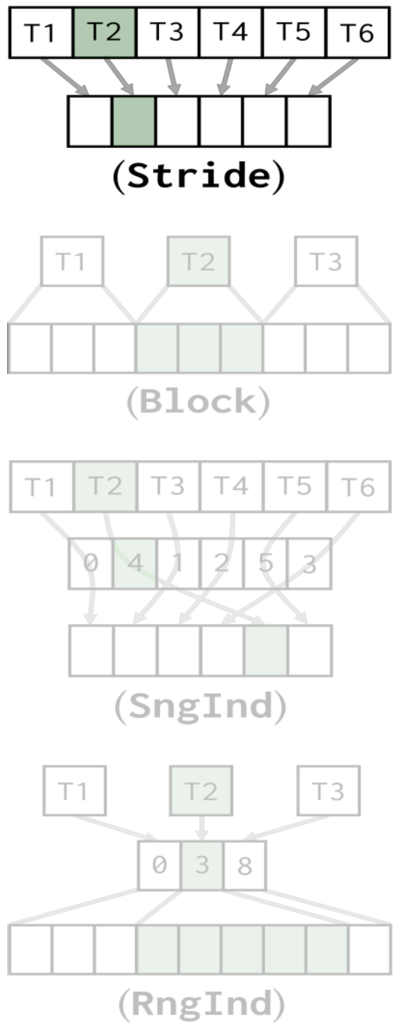
Fearless regular parallelism with Rust(+Rayon)

Regular parallelism:
- Known set of tasks
- Known dependences

```
fn par_increment(v: &mut [u32])
{
  v.par_iter_mut()
   .for_each(|vi|  *vi+=1);
}
```

stride pattern on **v**

task



Fearless regular parallelism with Rust(+Rayon)

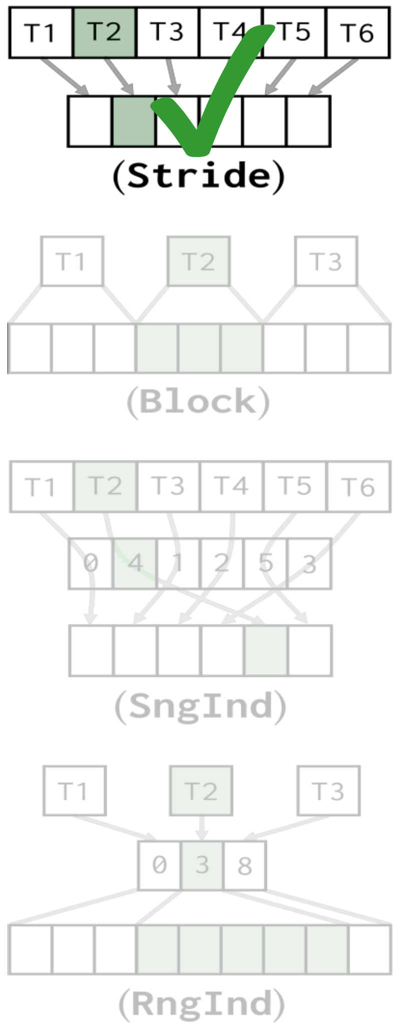
Regular parallelism:

- Known set of tasks
- Known dependences

```
fn par_increment(v: &mut [u32])
{
  v.par_iter_mut()
   .for_each(|vi|  *vi+=1);
}
```

stride pattern on **v**

task : Cannot access v ⇒ **No data races**



Fearless regular parallelism with Rust(+Rayon)

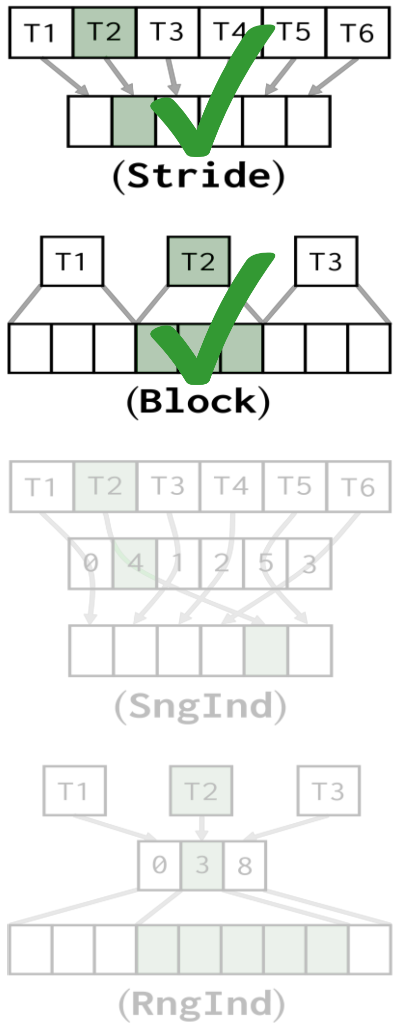
Regular parallelism:

- Known set of tasks
- Known dependences

```
fn par_increment(v: &mut [u32])
{
  v.par_iter_mut()
   .for_each(|vi|  *vi+=1);
}
```

stride pattern on **v**

task : Cannot access v $\Rightarrow$ **No data races**

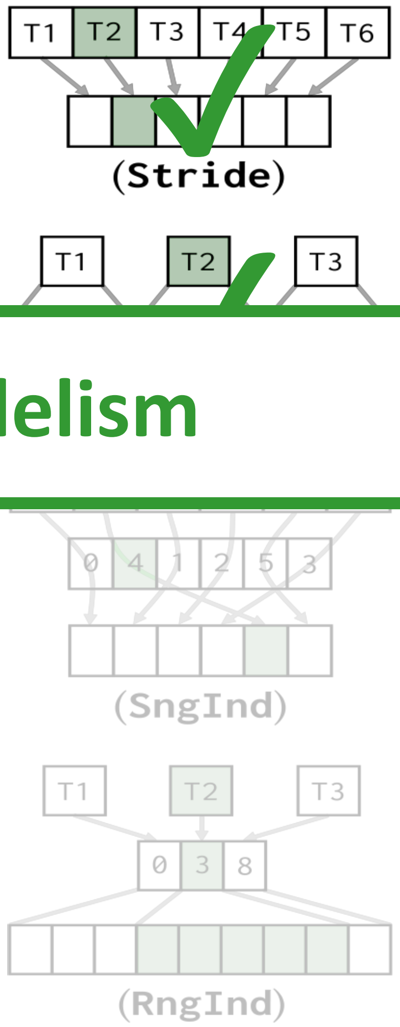


Fearless regular parallelism with Rust(+Rayon)

Regular parallelism:

Known set of tasks

Known dependences



Rust statically rules out data races for regular parallelism

```
{
  v.par_iter_mut()
   .for_each(|vi|  *vi+=1);
}
```

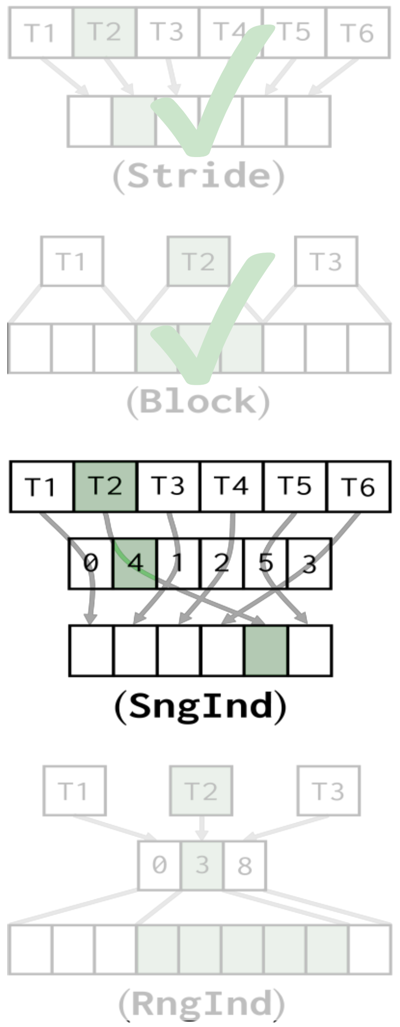
stride pattern on **v**

task : Cannot access v ⇒ **No data races**

Irregular parallelism remains scary

```
fn indirect_increment(v: &mut [u32], offsets: &[usize])
{
  (0..v.len()).into_par_iter()
    .for_each(|i|
      v[offsets[i]] += 1
    );
}
```

parallel loop

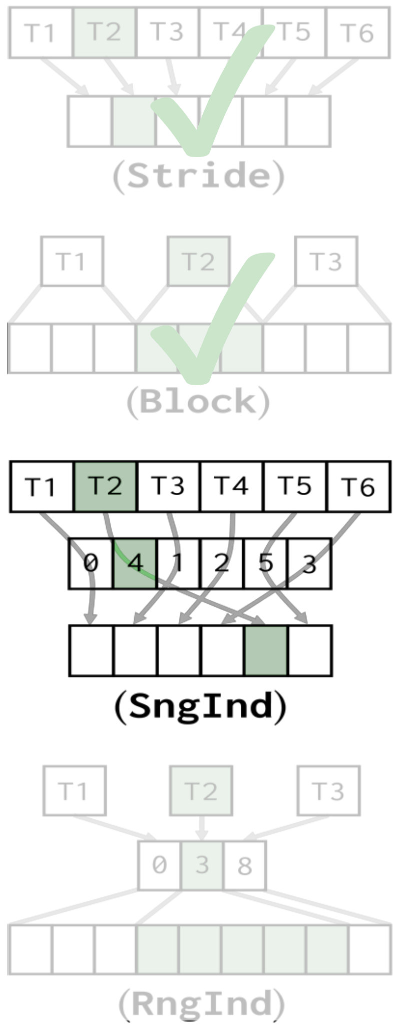


Irregular parallelism remains scary

```
fn indirect_increment(v: &mut [u32], offsets: &[usize])
{
  (0..v.len()).into_par_iter()
    .for_each(|i|
      v[offsets[i]] += 1
    );
}
```

parallel loop

Dangerous



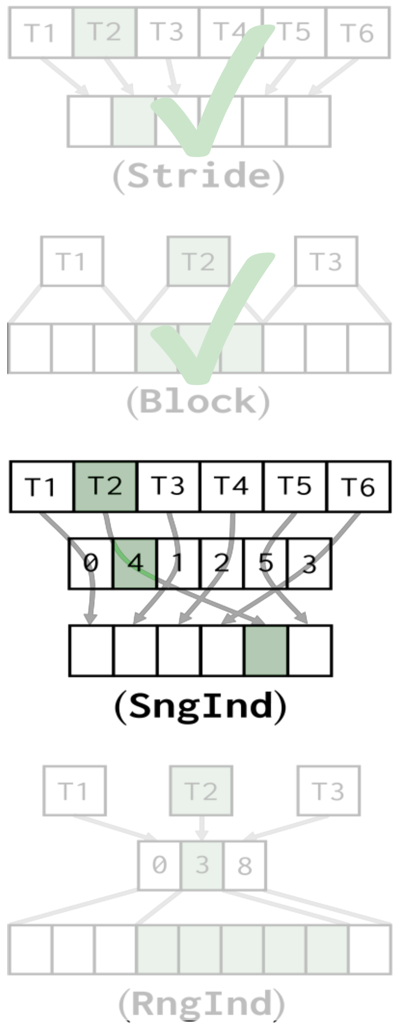
Irregular parallelism remains scary

```
fn indirect_increment(v: &mut [u32], offsets: &[usize])
{
  (0..v.len()).into_par_iter()
    .for_each(|i|
      v[offsets[i]] += 1
    );
}
```

parallel loop

~~Dangerous~~

Compile error



Irregular parallelism remains scary

```
fn indirect_increment(v: &mut [u32], offsets: &[usize])
{
  (0..v.len()).into_par_iter()
    .for_each(|i|
      v[offsets[i]] += 1
    );
}
```

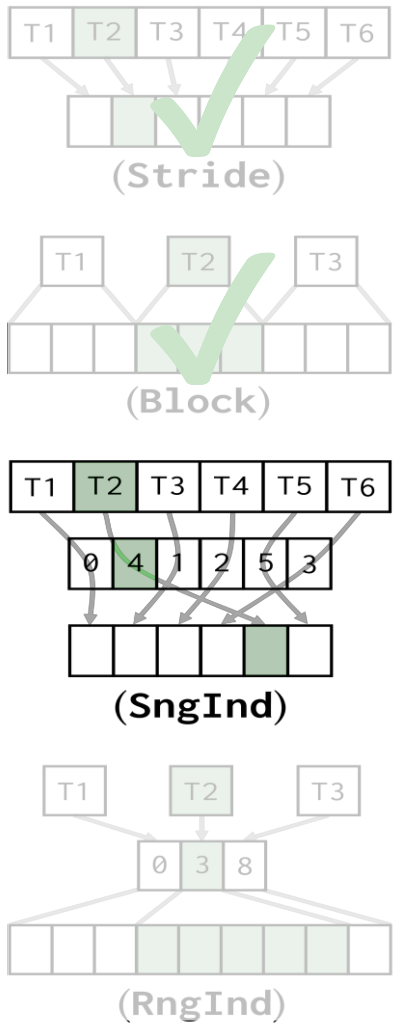
parallel loop

~~Dangerous~~

Compile error

offsets

Duplicates: Synchronization 😱



Irregular parallelism remains scary

```
fn indirect_increment(v: &mut [u32], offsets: &[usize])
{
  (0..v.len()).into_par_iter()
    .for_each(|i|
      v[offsets[i]] += 1
    );
}
```

parallel loop

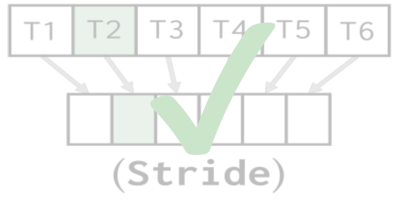
~~Dangerous~~

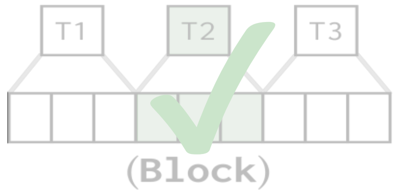
Compile error

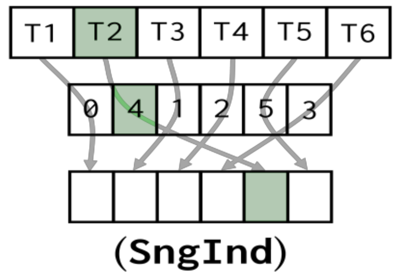
offsets

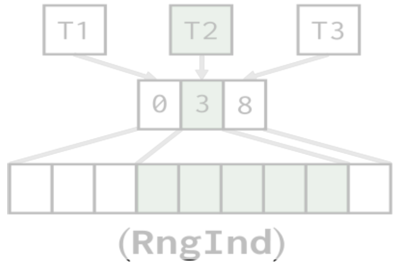
Duplicates: Synchronization 😱

Unique
- Unsafe without checks 😱
- Unsafe with checks 🐌
- Synchronization 😱 🐌









Irregular parallelism remains scary

```
fn indirect_increment(v: &mut [u32], offsets: &[usize])
{
  (0..v.len()).into_par_iter()
    .for_each(|i|
      v[offsets[i]] += 1
    );
}
```

parallel loop

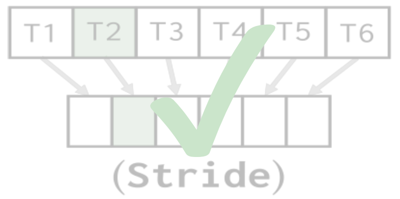
~~Dangerous~~

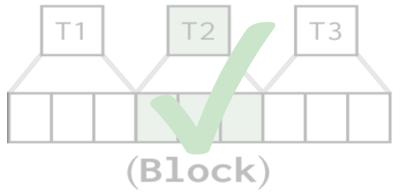
Compile error

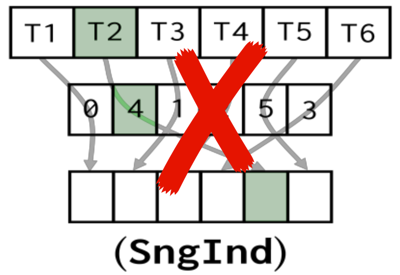
offsets

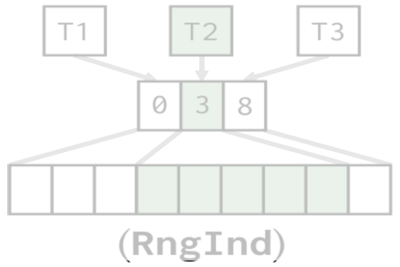
Duplicates: Synchronization 😱

Unique
- Unsafe without checks 😱
- Unsafe with checks 🐌
- Synchronization 😱 🐌









Irregular parallelism remains scary

```
fn indirect_increment(v: &mut [u32], offsets: &[usize])
{
  (0..v.len()).into_par_iter()
    .for_each(|i|
      v[offsets[i]] += 1
    );
}
```

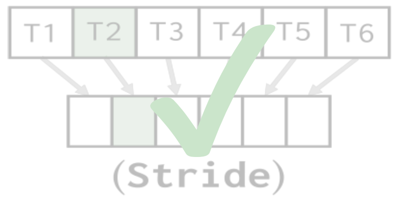
parallel loop

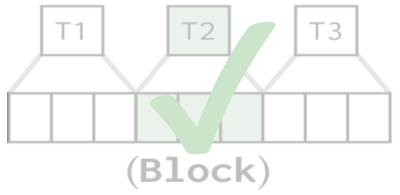
~~Dangerous~~

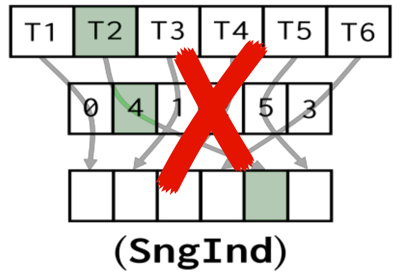
Compile error

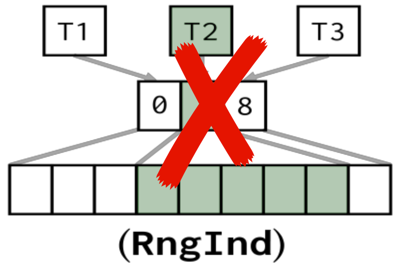
offsets

- Duplicates: Synchronization 😱
- Unique
  - Unsafe without checks 😱
  - Unsafe with checks 🐌
  - Synchronization 😱 🐌







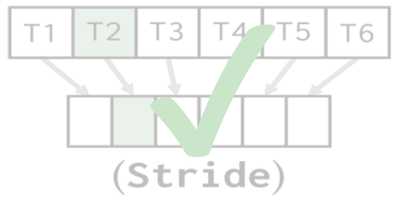


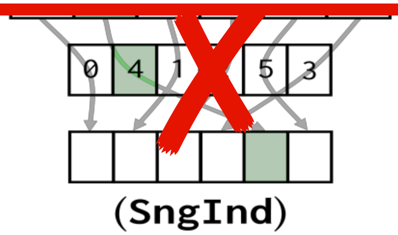
Irregular parallelism remains scary

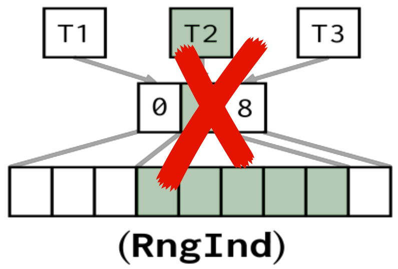
```
fn indirect_increment(v: &mut [u32], offsets: &[usize])
{
  (0..v.len()).into_par_iter() ← parallel loop
     .for_each(|i|
}
```

Rust solutions for irregular parallelism are not fearless

offsets
- Duplicates: Synchronization 😱
- Unique
  - Unsafe without checks 😱
  - Unsafe with checks 🐌
  - Synchronization 😱 🐌



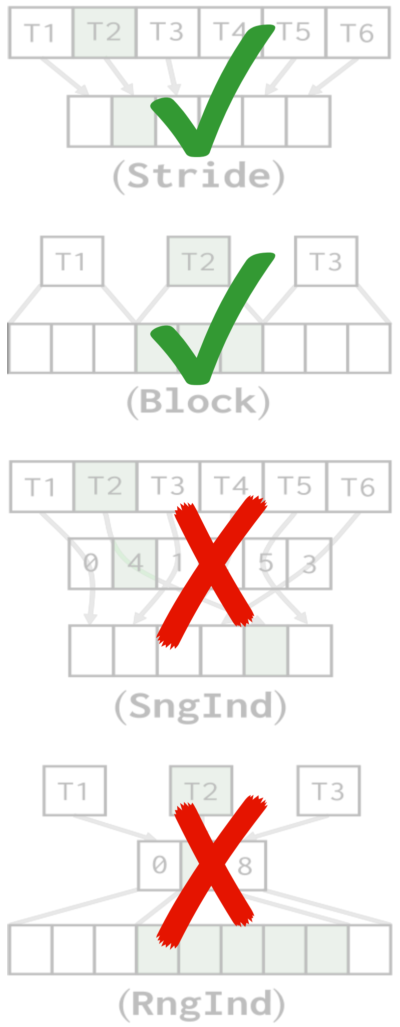




Does this matter?
Irregular parallelism is common in PBBS!

Regular parallelism ✓

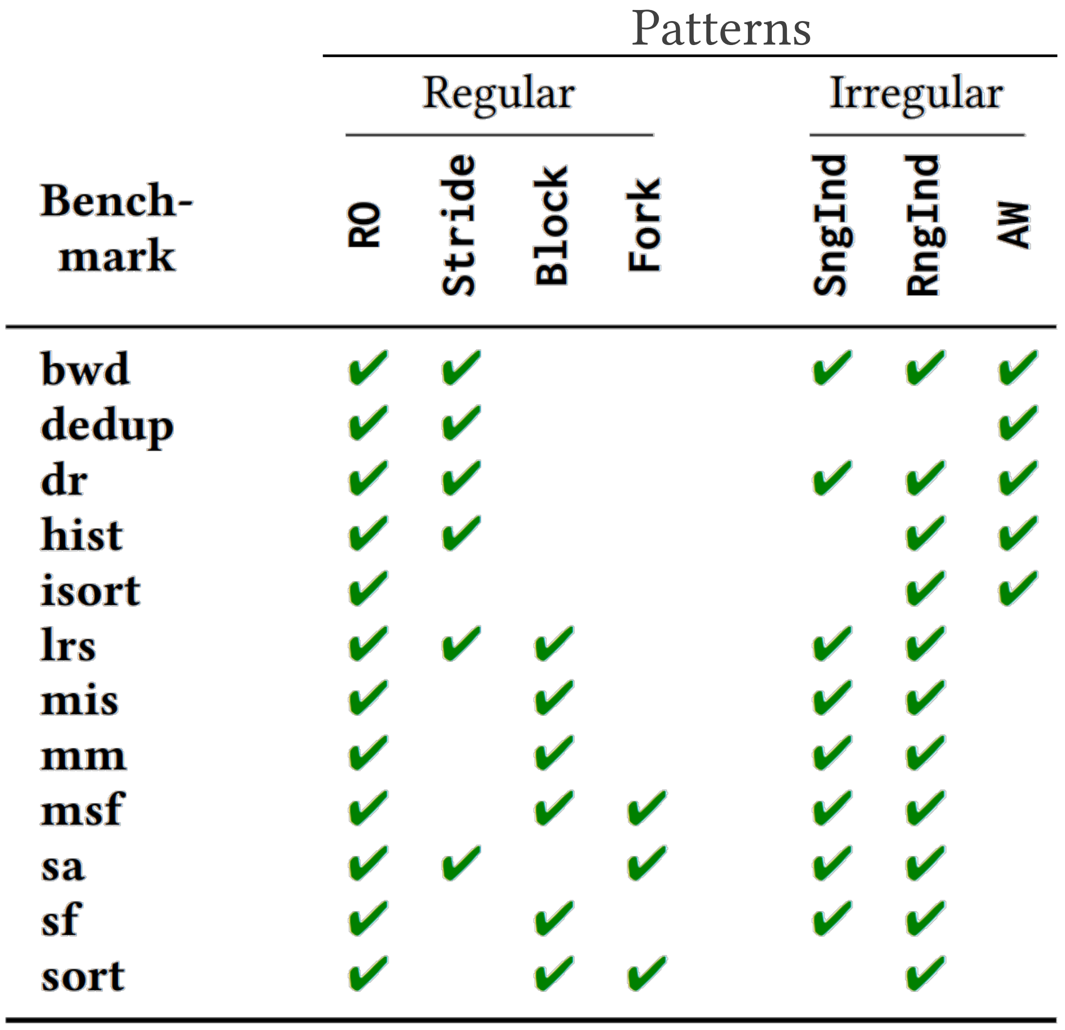
Irregular parallelism ✗



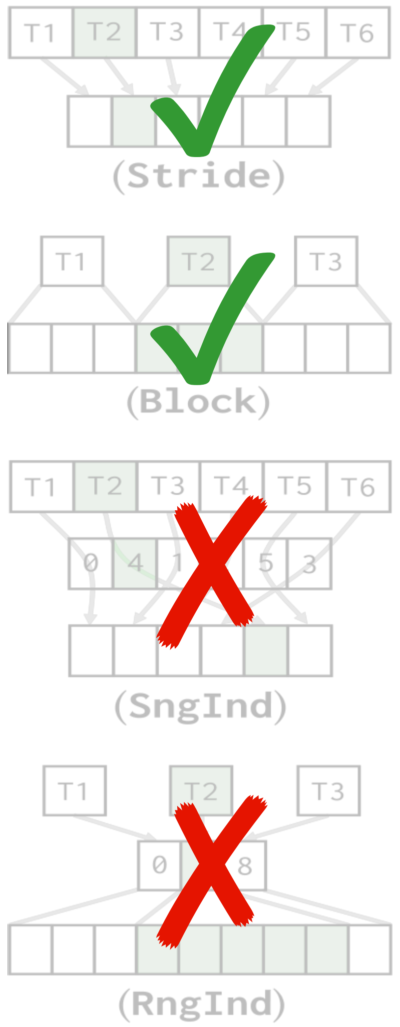
Does this matter?
Irregular parallelism is common in PBBS!

Regular parallelism ✓

Irregular parallelism ✗



	Patterns						
	Regular				Irregular		
Bench-mark	RO	Stride	Block	Fork	SngInd	RngInd	AW
bwd	✓	✓			✓	✓	✓
dedup	✓	✓					✓
dr	✓	✓			✓	✓	✓
hist	✓	✓				✓	✓
isort	✓					✓	✓
lrs	✓	✓	✓		✓	✓	
mis	✓		✓		✓	✓	
mm	✓		✓		✓	✓	
msf	✓		✓	✓	✓	✓	
sa	✓	✓		✓	✓	✓	
sf	✓		✓		✓	✓	
sort	✓		✓	✓		✓	



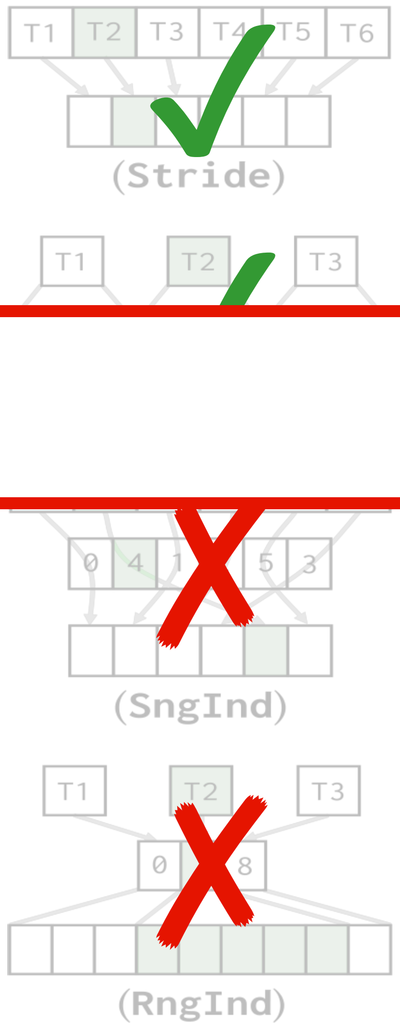
Does this matter?
Irregular parallelism is common in PBBS!

Regular parallelism ✓

Irregular parallelism ✗

Expressing PBBS in Rust is not fearless

	Patterns						
	Regular				Irregular		
Bench-mark	RO	Stride	Block	Fork	SngInd	RngInd	AW
hist	✓	✓				✓	✓
isort	✓					✓	✓
lrs	✓	✓	✓		✓	✓	
mis	✓		✓		✓	✓	
mm	✓		✓		✓	✓	
msf	✓		✓	✓	✓	✓	
sa	✓	✓		✓	✓	✓	
sf	✓		✓		✓	✓	
sort	✓		✓	✓		✓	



Conclusions

Regular parallelism

Irregular parallelism

Easy parallelism is fearless!

Hard parallelism is still scary...

github.com/mcj-group/rusty-pbbs