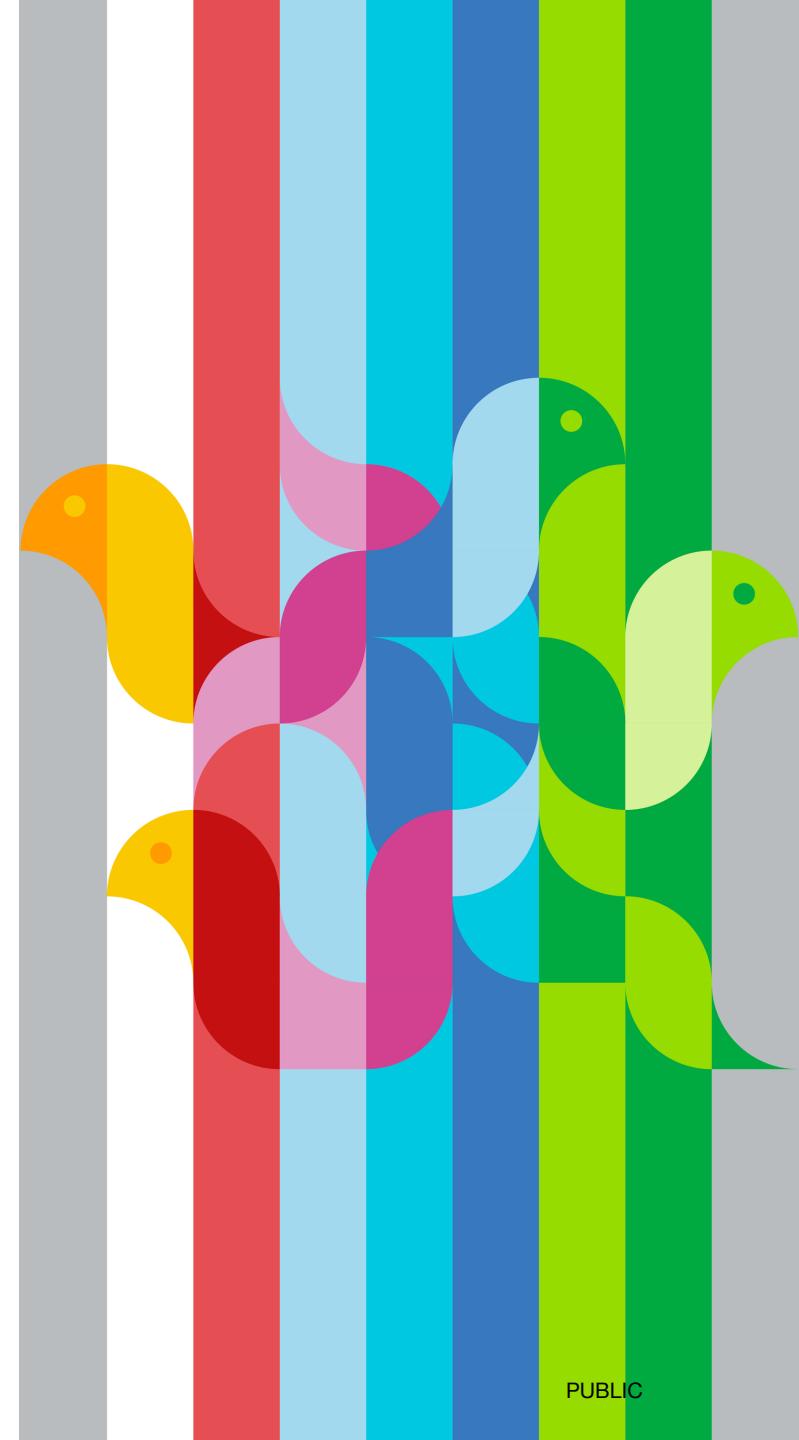


Rustifying Python: A Practical Guide to Achieving High Performance While Maintaining Observability

Max Höhl
SAP SE



“

**Don't force me to
deal with your shiny
language of the day.
Maintaining multi-
language projects is
a pain**

- Christoph Hellwig



/whme/PyConDE-2025

PUBLIC

- **How to identify parts suitable for Rust**



- **How to identify parts suitable for Rust**
- **How to migrate to Rust and deploy to production**



- **How to identify parts suitable for Rust**
- **How to migrate to Rust and deploy to production**
- **Lessons learned**



- **How to identify parts suitable for Rust**
- **How to migrate to Rust and deploy to production**
- **Lessons learned**











1 – Conflict



Cluster Utilization



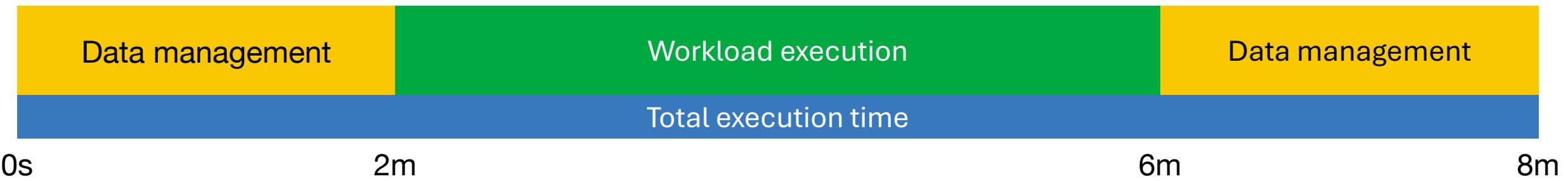
Data management

1 – Conflict

Cluster Utilization



1 – Conflict



2 – Cluster Utilization

```
get_resource_offers_from_cluster
```

2 – Cluster Utilization

get_resource_offers_from_cluster

getprocessable_tasks

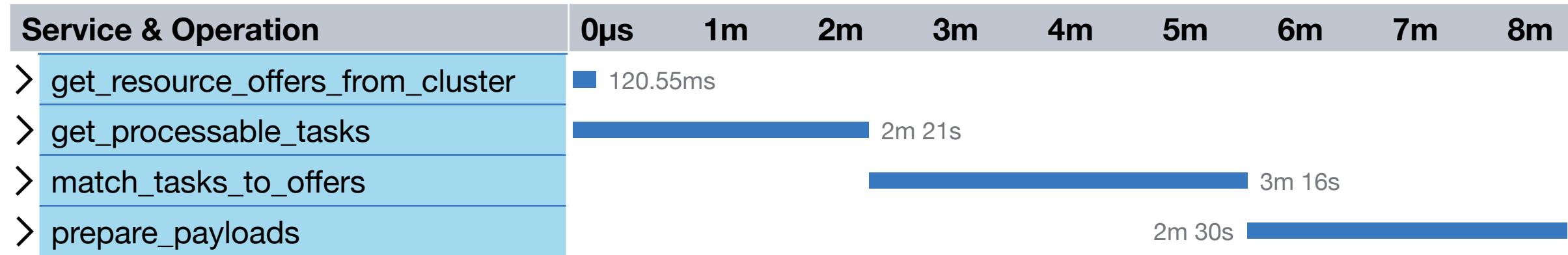
2 – Cluster Utilization

```
get_resource_offers_from_cluster  
getprocessable_tasks  
match_tasks_to_offers
```

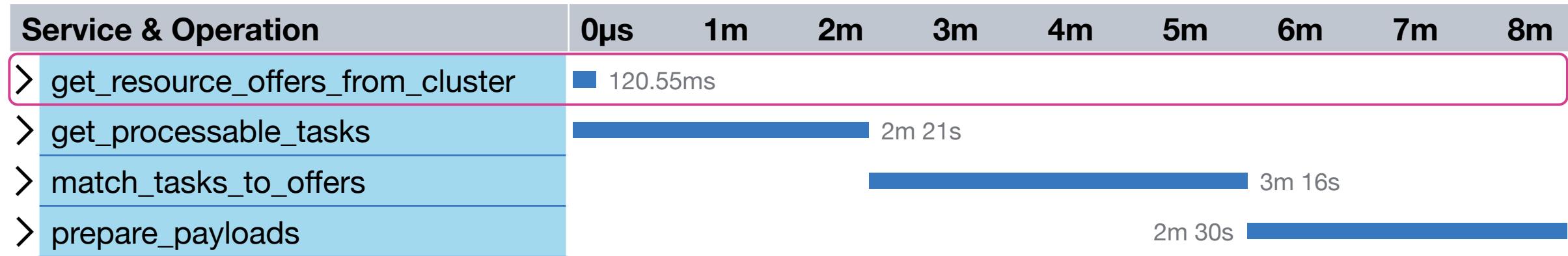
2 – Cluster Utilization

```
get_resource_offers_from_cluster  
getprocessable_tasks  
match_tasks_to_offers  
prepare_payloads
```

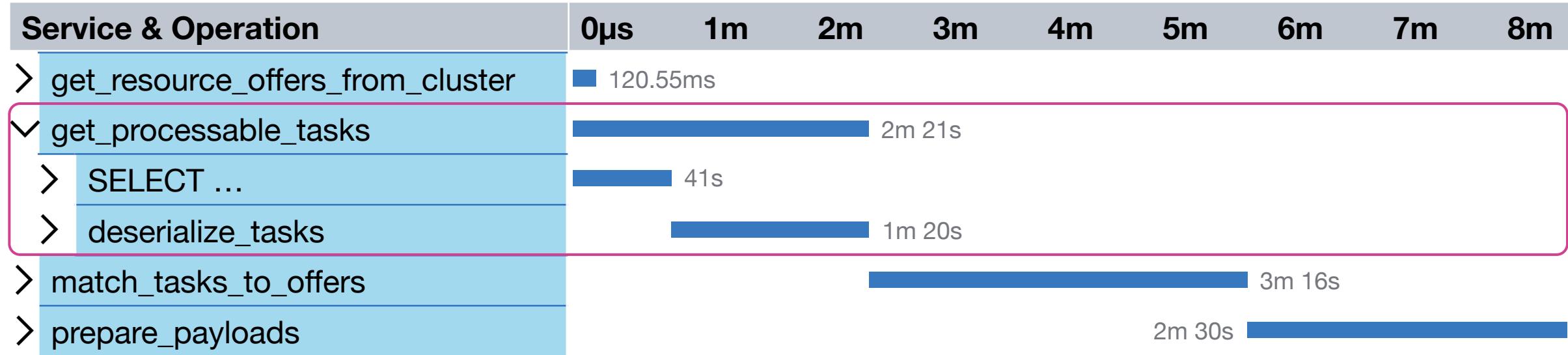
2 – Cluster Utilization



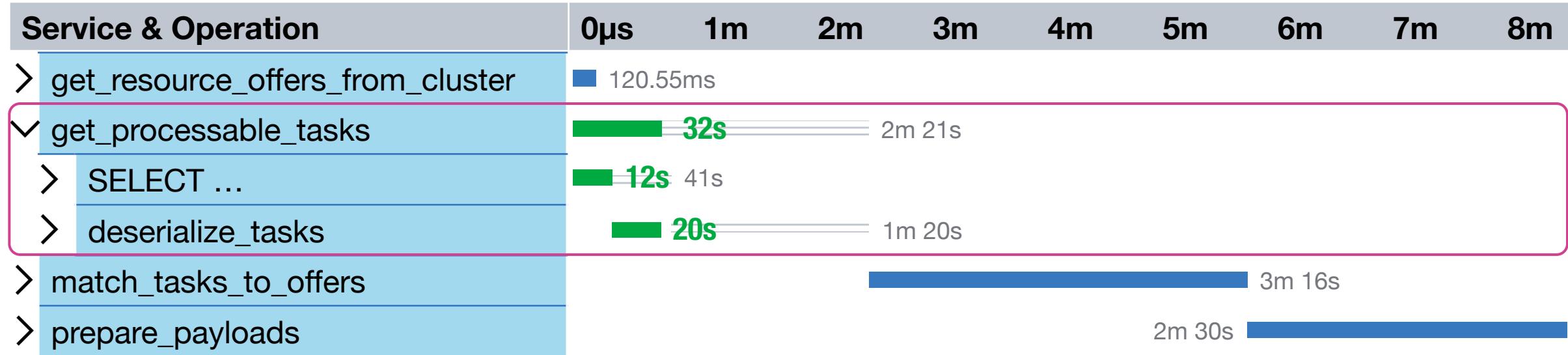
2 – Cluster Utilization



2 – Cluster Utilization



2 – Cluster Utilization

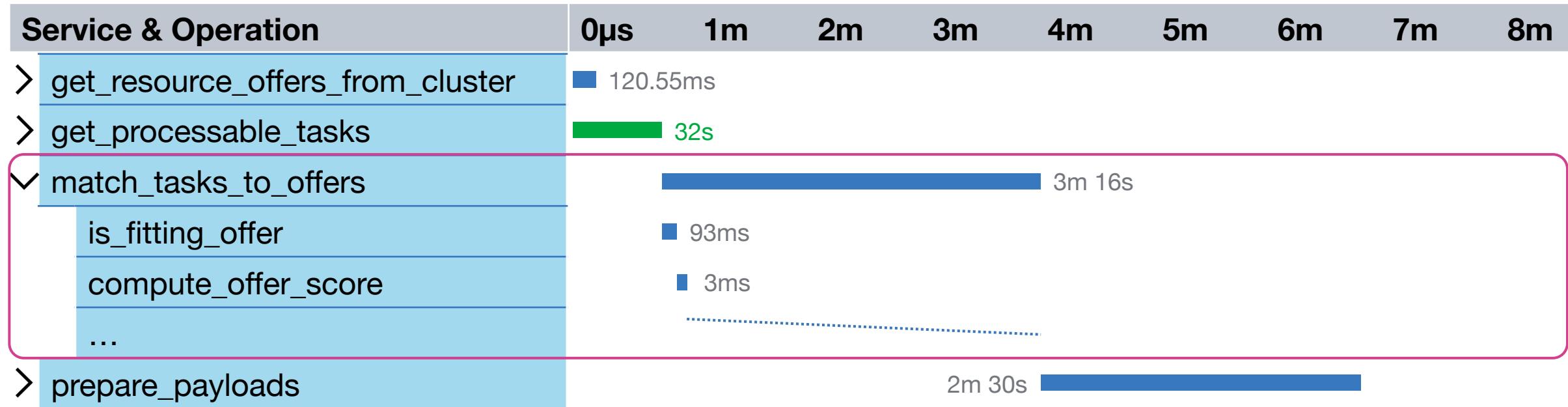


2 – Cluster Utilization

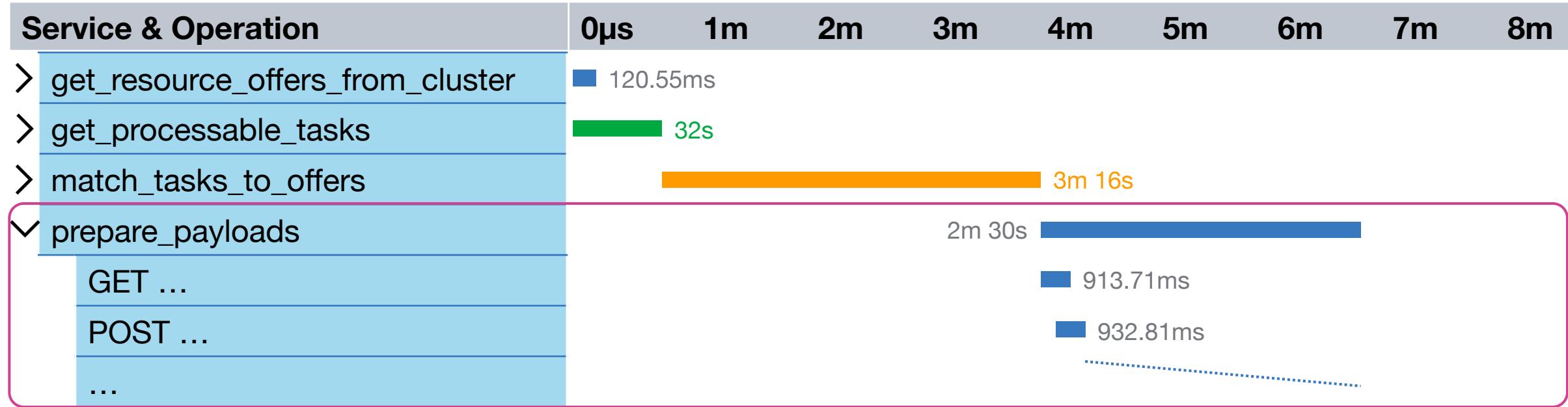
Silver bullet

Focus on understanding and fixing the underlying root cause

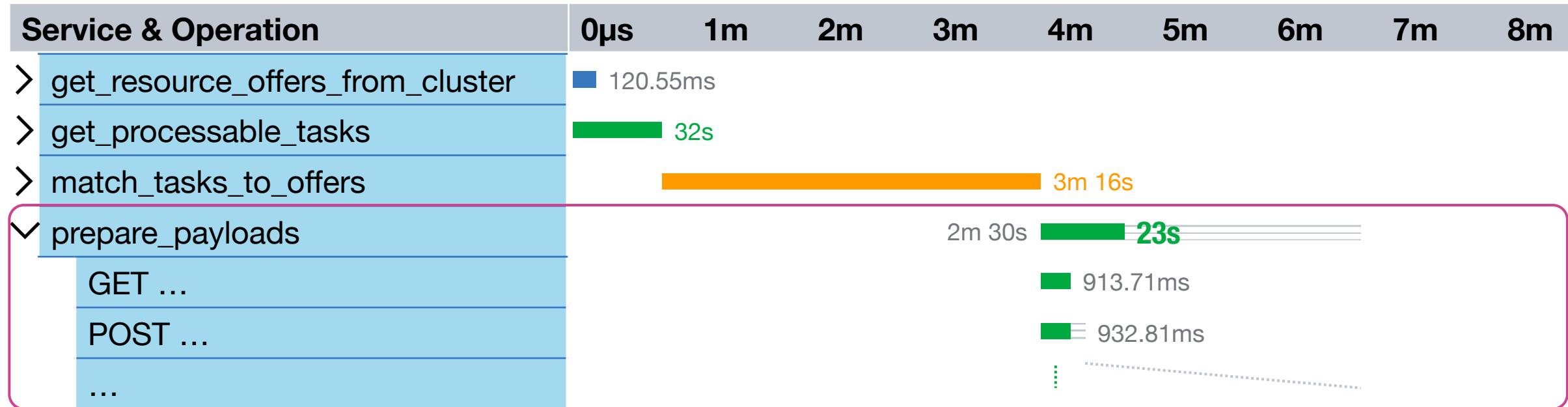
2 – Cluster Utilization



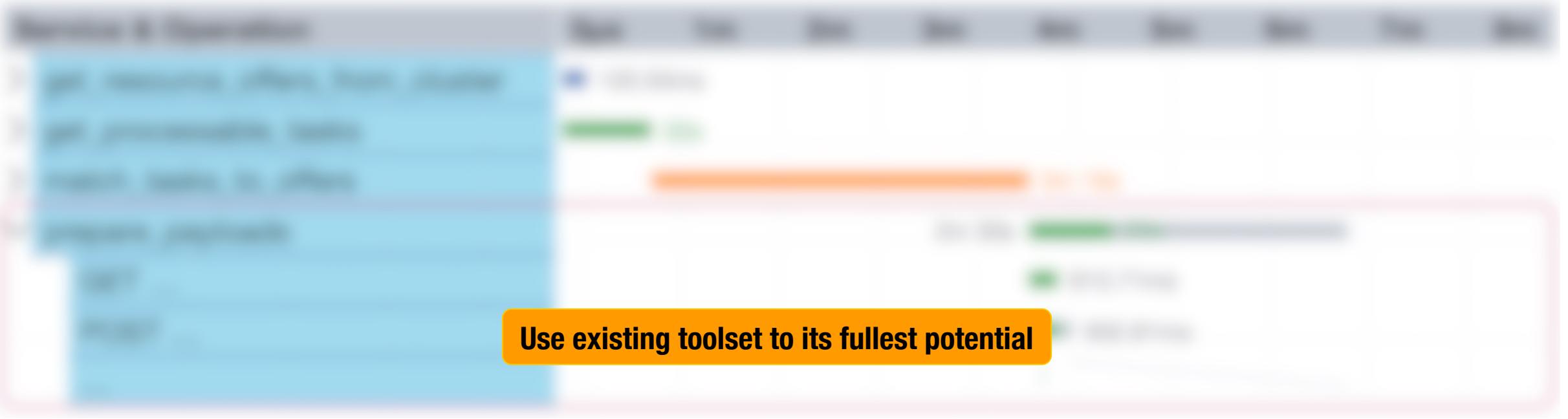
2 – Cluster Utilization



2 – Cluster Utilization



2 – Cluster Utilization



Use existing toolset to its fullest potential

3 – Offer matching

```
11  def match_tasks_to_offers(  
12      tasks: list[Task], offers: list[Offer]  
13  ) -> dict[OfferName, TaskName]:  
14      """Returns a mapping from offer name to task name where the key is the name  
15      of the bestfit offer for the task.  
16      """  
17  
18      result: dict[OfferName, TaskName] = {}  
19  
20      for task in tasks:  
21          best_offer_name: OfferName | None, score: float | None, _ = best_fit(task, offers)  
22          if best_offer_name is not None:  
23              result[best_offer_name] = task  
24  
25      return result
```

3 – Offer matching

```
11 def match_tasks_to_offers(  
12     tasks: list[Task], offers: list[Offer]  
13 ) -> dict[OfferName, TaskName]:  
14     """Returns a mapping from offer name to task name where the key is the name  
15     of the bestfit offer for the task.  
16     """  
17     result = {}  
18     for task in tasks:  
19         best_index_score: tuple[OfferIndex | None, float] = (None, 31.0)  
20         for index, offer in enumerate(offers):  
21             if is_fitting_offer(task, offer):  
22                 score = compute_score(task, offer)  
23                 if score < best_index_score[1]:  
24                     best_index_score = (index, score)
```

3 – Offer matching

```
11  def match_tasks_to_offers(  
12      tasks: list[Task], offers: list[Offer]  
13  ) -> dict[OfferName, TaskName]:  
14      """Returns a mapping from offer name to task name where the key is the name  
15      of the bestfit offer for the task.  
16      """  
17      result = {}  
18      for task in tasks:  
19          best_index_score: tuple[OfferIndex | None, float] = (None, 31.0)  
20          for index, offer in enumerate(offers):  
21              if is_fitting_offer(task, offer):  
22                  score = compute_score(task, offer)  
23                  if score < best_index_score[1]:  
24                      best_index_score = (index, score)  
25                  if (best_index := best_index_score[0]) is not None:  
26                      result[swap_remove(offers, best_index).name] = task.name  
27  
28      return result
```

3 – Offer matching

Implementation	Total time	is_fitting_offer time	compute_score time
Production benchmark	3m16s	93ms	3ms
Python single threaded	2s	83ms	2ms

3 – Offer matching

```
128     def match_tasks_to_offers(  
129         pool_class: type[Pool], tasks: list[Task], offers: list[Offer]  
130     ) -> dict[OfferName, TaskName]:  
131         """Match the given tasks to the given offers"""\n132         pool = pool_class(os.cpu_count())\n\n133         result: dict[OfferName, TaskName] = {}\n\n134         for task in tasks:  
135             best_index_score: tuple[OfferName | None, float] = (None, 0.0)\n\n136             for offer, score in enumerate(offers):\n\n137                 pool.starmap(get_offer_score, [(task, offer)])\n\n138                 if score > best_index_score[1]:\n\n139                     best_index_score = (offer, score)\n\n140             if best_index_score[0] is None:\n\n141                 continue\n\n142             result[best_index_score[0]] = task.name\n\n143         pool.close()\n\n144         pool.join()\n\n145         return result
```

3 – Offer matching

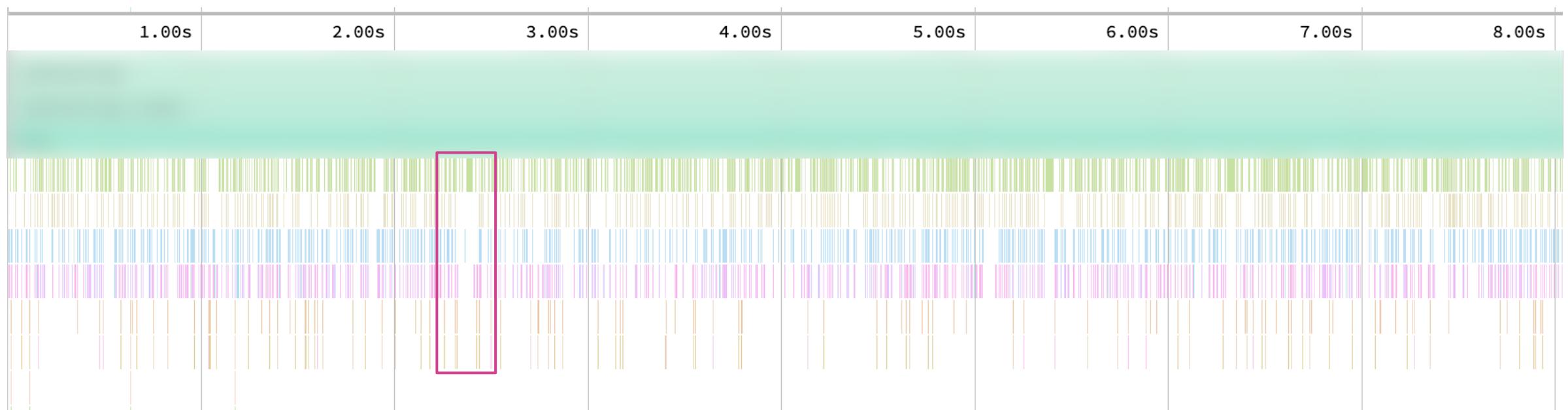
```
128     def match_tasks_to_offers(
129         pool_class: type[Pool], tasks: list[Task], offers: list[Offer]
130     ) -> dict[OfferName, TaskName]:
131         """Match the given tasks to the given offers"""
132         pool = pool_class(os.cpu_count())
133         result = {}
134         for task in tasks:
135             best_index_score: tuple[OfferIndex | None, float] = (None, 31.0)
136             for index, score in enumerate(
137                 pool.starmap(get_offer_score, [(task, offer) for offer in offers])
138             ):
139                 if score is not None and score < best_index_score[1]:
140                     best_index_score = (index, score)
141             if best_index_score[0] is None:
142                 continue
143             result[(swap_remove(offers, best_index_score[0]).name)] = task.name
144         pool.close()
145         pool.join()
146         return result
```

3 – Offer matching

Implementation	Total time	is_fitting_offer time	compute_score time
Production benchmark	3m16s	93ms	3ms
Python single threaded	2s	83ms	2ms
Python multi threaded	27s	495ms	29ms

3 – Offer matching

Implementation	Total time	is_fitting_offer time	compute_score time
Production benchmark	3m16s	93ms	3ms
Python single threaded	2s	83ms	2ms
Python multi threaded	27s	495ms	29ms

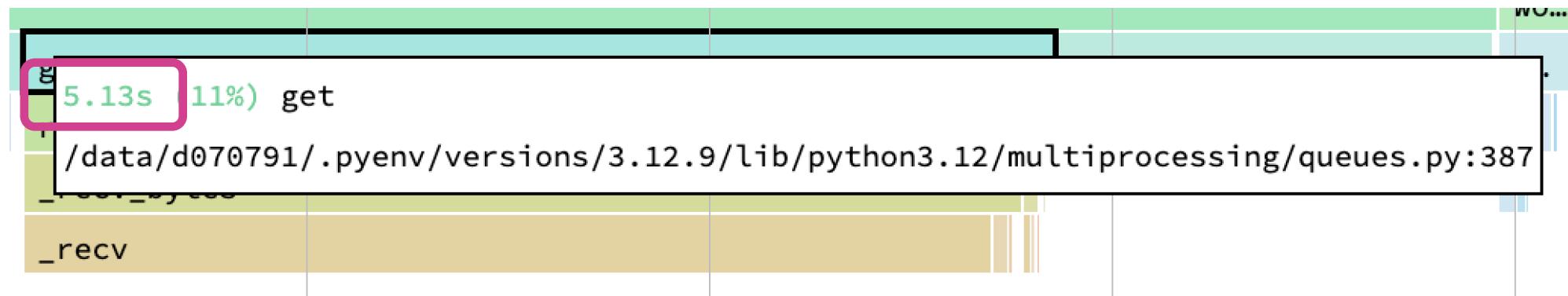


3 – Offer matching

Implementation	Total time	is_fitting_offer time	compute_score time
Production benchmark	3m16s	93ms	3ms
Python single threaded	2s	83ms	2ms
Python multi threaded	27s	495ms	29ms
Python multi processing	70s	50ms	2ms

3 – Offer matching

Implementation	Total time	is_fitting_offer time	compute_score time
Production benchmark	3m16s	93ms	3ms
Python single threaded	2s	83ms	2ms
Python multi threaded	27s	495ms	29ms
Python multi processing	70s	50ms	2ms



3 – Offer matching

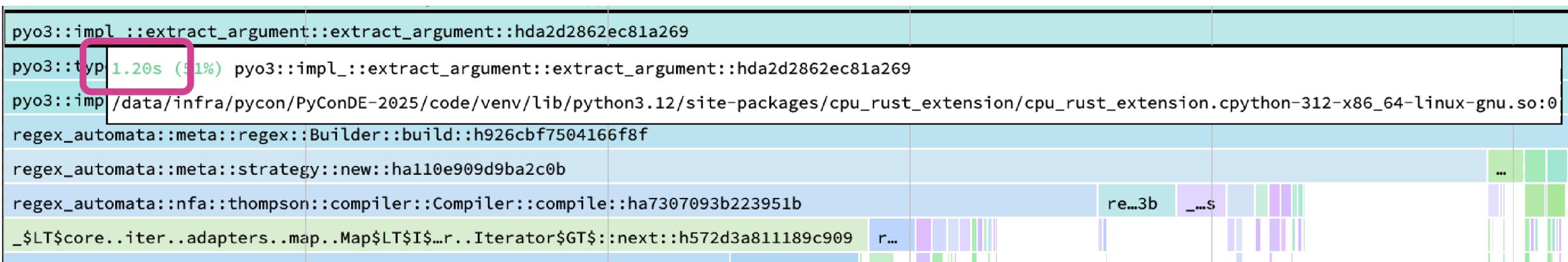
```
11 fn match_tasks_to_offers(tasks: Vec<Task>, mut offers: Vec<Offer>) -> HashMap<String, String> {
12     let mut result = HashMap::with_capacity(offers.len());
13     let mut best_index_score: (Option<usize>, f64);
14     for task in tasks.iter() {
15         best_index_score = (None, 31.0);
16         for (index, offer) in offers.iter().enumerate() {
17             if is_fitting_offer(task, offer) {
18                 let score = compute_score(task, offer);
19                 if score < best_index_score.1 {
20                     best_index_score = (Some(index), score);
21                 }
22             }
23         }
24         if let Some(best_index) = best_index_score.0 {
25             result.insert(offers.swap_remove(best_index).name, task.name.clone());
26         }
27     }
28     result
29 }
```

3 – Offer matching

Implementation	Total time	is_fitting_offer time	compute_score time
Production benchmark	3m16s	93ms	3ms
Python single threaded	2s	83ms	2ms
Python multi threaded	27s	495ms	29ms
Python multi processing	70s	50ms	2ms
Rust single threaded	2s	20ms	150µs

3 – Offer matching

Implementation	Total time	is_fitting_offer time	compute_score time
Production benchmark	3m16s	93ms	3ms
Python single threaded	2s	83ms	2ms
Python multi threaded	27s	495ms	29ms
Python multi processing	70s	50ms	2ms
Rust single threaded	2s	20ms	150µs



3 – Offer matching

Implementation	Total time	is_fitting_offer time	compute_score time
Production benchmark	3m16s	93ms	3ms
Python single threaded	2s	83ms	2ms
Python multi threaded	27s	495ms	29ms
Python multi processing	70s	50ms	2ms
Rust single threaded	2s	20ms	150µs
Rust multi threaded	3s	20ms	150µs

3 – Offer matching

Implementation	Total time	is_fitting_offer time	compute_score time
	10K Tasks 2K Offers	20K Tasks 20K Offers	
Production benchmark	-	3m16s	93ms
Python single threaded	2s	83ms	2ms
Python multi threaded	27s	495ms	29ms
Python multi processing	70s	50ms	2ms
Rust single threaded	2s	20ms	150µs
Rust multi threaded	3s	20ms	150µs

3 – Offer matching

Implementation	Total time		is_fitting_offer time	compute_score time
	10K Tasks 2K Offers	20K Tasks 20K Offers		
Production benchmark	-	3m16s	93ms	3ms
Python single threaded	2s	3m21s	83ms	2ms
Python multi threaded	27s	6m40s	495ms	29ms
Python multi processing	70s	20m12s	50ms	2ms
Rust single threaded	2s	6s	20ms	150µs
Rust multi threaded	3s	4s	20ms	150µs

3 – Offer matching

Implementation	Total time		is_fitting_offer time	compute_score time
	10K Tasks 2K Offers	20K Tasks 20K Offers		
Production benchmark	-	3m16s	93ms	3ms
Python single threaded	2s	3m21s	83ms	2ms
Python multi threaded	27s	6m40s	495ms	29ms
Python multi processing	70s	20m12s	50ms	2ms
Rust single threaded	2s	6s x30	20ms	150µs
Rust multi threaded	3s	4s x49	20ms	150µs

3 – Offer matching

Implementation	Total time		is_fitting_offer time	compute_score time
	10K Tasks 2K Offers	20K Tasks 20K Offers		
Production benchmark	-	3m16s	93ms	3ms
Python single threaded	2s	3m21s x100	83ms	2ms
Python multi threaded	27s	6m40s	495ms	29ms
Python multi processing	70s	20m12s	50ms	2ms
Rust single threaded	2s	6s x3	20ms	150µs
Rust multi threaded	3s	4s x2	20ms	150µs

3 – Offer matching

Generate different test data (size, structure, etc.)

Scale matters – Rust integration comes at a cost

3 – Offer matching

Implementation	Total time (+increase)	Background thread operations/ms
Python single threaded	3m51s (+ 30s)	61
Python multi threaded	7m40s (+ 1m)	51
Python multi processing	20m12s (+ 0s)	79
Rust single threaded	6s (+ 0s)	0
Rust multi threaded	4s (+ 0s)	0

3 – Offer matching

Implementation	Total time (+increase)	Background thread operations/ms
Python single threaded	3m51s (+ 30s)	61
Python multi threaded	7m40s (+ 1m)	51
Python multi processing	20m12s (+ 0s)	79
Rust single threaded	6s (+ 0s)	0
Rust multi threaded	4s (+ 0s)	0

3 – Offer matching

Implementation	Total time (+increase)	Background thread operations/ms
Python single threaded	3m51s (+ 30s)	61
Python multi threaded	7m40s (+ 1m)	51
Python multi processing	20m12s (+ 0s)	79
Rust single threaded	6s (+ 0s)	0
Rust multi threaded	4s (+ 0s)	0

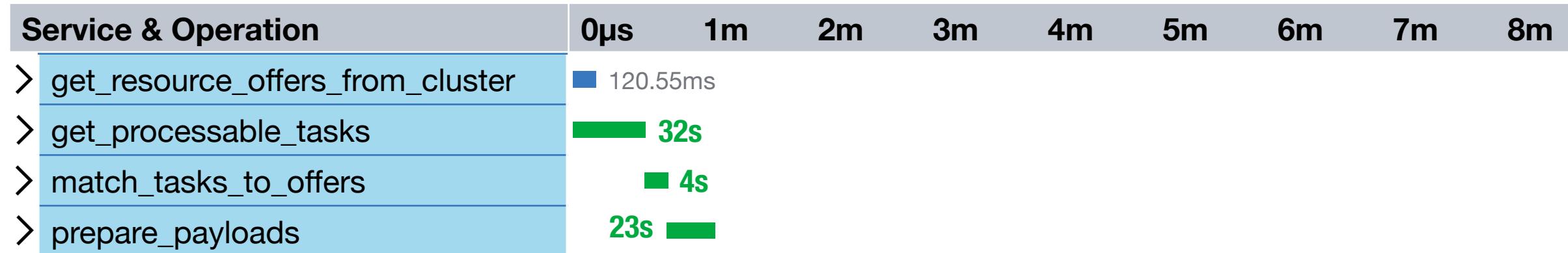
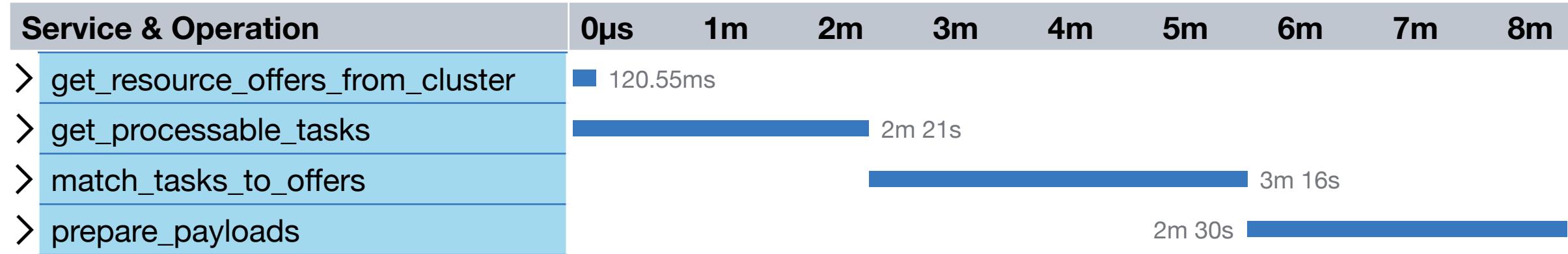
```
32 32 #[pyfunction]
33 33 #[pyo3(name = "match_single_threaded")]
34 pub fn main(tasks: Vec<Task>, offers: Vec<Offer>) -> HashMap<String, String> {
35     match_tasks_to_offers(tasks, offers)
34 pub fn main(py: Python, tasks: Vec<Task>, offers: Vec<Offer>) -> HashMap<String, String> {
35     py.allow_threads(move || match_tasks_to_offers(tasks, offers))
36 36 }
```

3 – Offer matching

Implementation	Total time (+increase)	Background thread operations/ms
Python single threaded	3m51s (+ 30s)	61
Python multi threaded	7m40s (+ 1m)	51
Python multi processing	20m12s (+ 0s)	79
Rust single threaded	6s (+ 50ms)	80
Rust multi threaded	4s (+ 50ms)	79

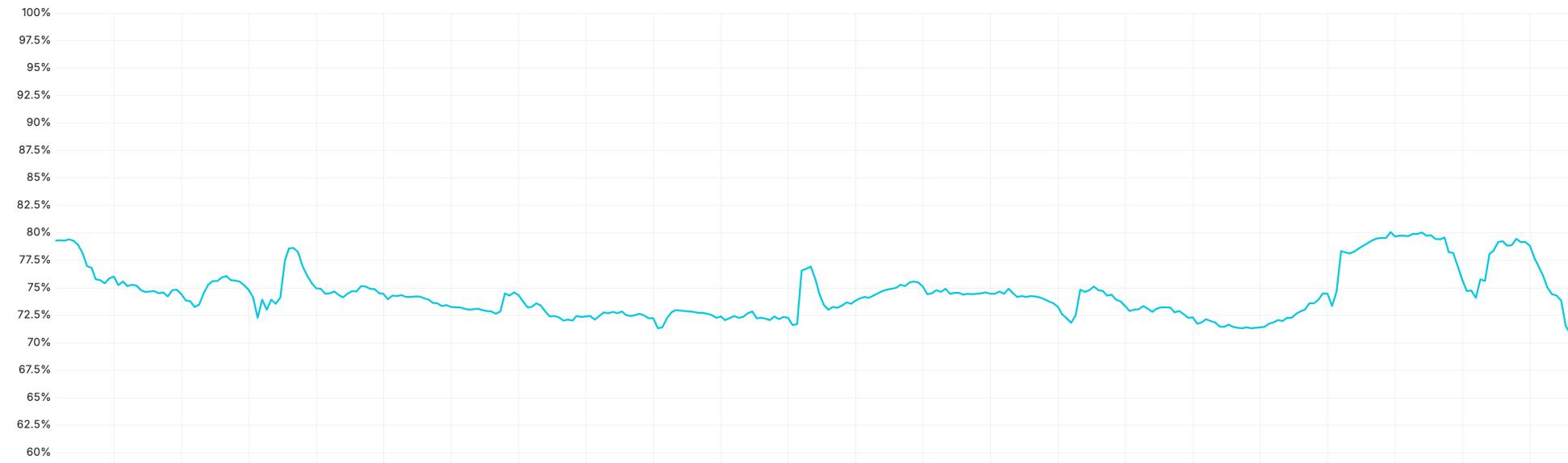
```
32  32 #[pyfunction]
33  33 #[pyo3(name = "match_single_threaded")]
34  pub fn main(tasks: Vec<Task>, offers: Vec<Offer>) -> HashMap<String, String> {
35      match_tasks_to_offers(tasks, offers)
34  pub fn main(py: Python, tasks: Vec<Task>, offers: Vec<Offer>) -> HashMap<String, String> {
35      py.allow_threads(move || match_tasks_to_offers(tasks, offers))
36  }
```

3 – Offer matching



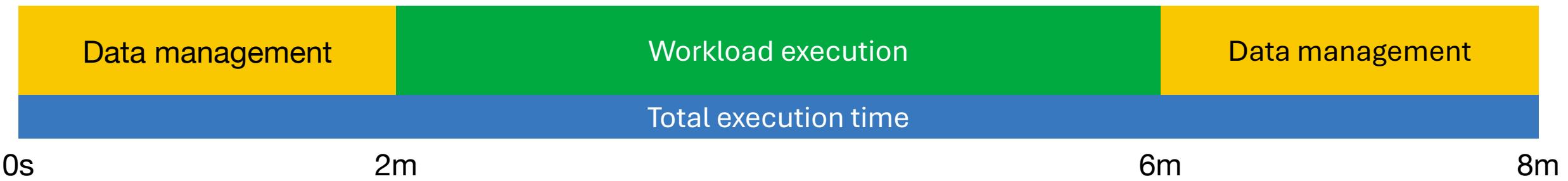
3 – Offer matching

Cluster Utilization



Service & Operation	0μs	1m	2m	3m	4m	5m	6m	7m	8m
> get_resource_offers_from_cluster	120.55ms								
> getprocessable_tasks		32s							
> match_tasks_to_offers			4s						
> prepare_payloads	23s								

4 – Data management overhead



4 – Data management overhead

"External factor"	Used	Available
Local disk throughput	482 MB/s	1.2 GB/s
Cloud storage throughput	5 GB/s	>100 GB/s
Network bandwidth	5 GB/s	10 GB/s

4 – Data management overhead

"External factor"	Used	Available
Local disk throughput	482 MB/s	1.2 GB/s
Cloud storage throughput	5 GB/s	>100 GB/s
Network bandwidth	5 GB/s	10 GB/s

```
6  def main(source: str, target: str) -> None:
7      """Copy source to target path"""
8      with open(source, "rb") as infile, open(target, "wb") as outfile:
9          while buffer := infile.read(16 * MiB):
10              outfile.write(buffer)
```

4 – Data management overhead

Implementation	Throughput
Python	682 MB/s
Rust	689 MB/s

```
6  def main(source: str, target: str) -> None:
7      """Copy source to target path"""
8      with open(source, "rb") as infile, open(target, "wb") as outfile:
9          while buffer := infile.read(16 * MiB):
10              outfile.write(buffer)
```

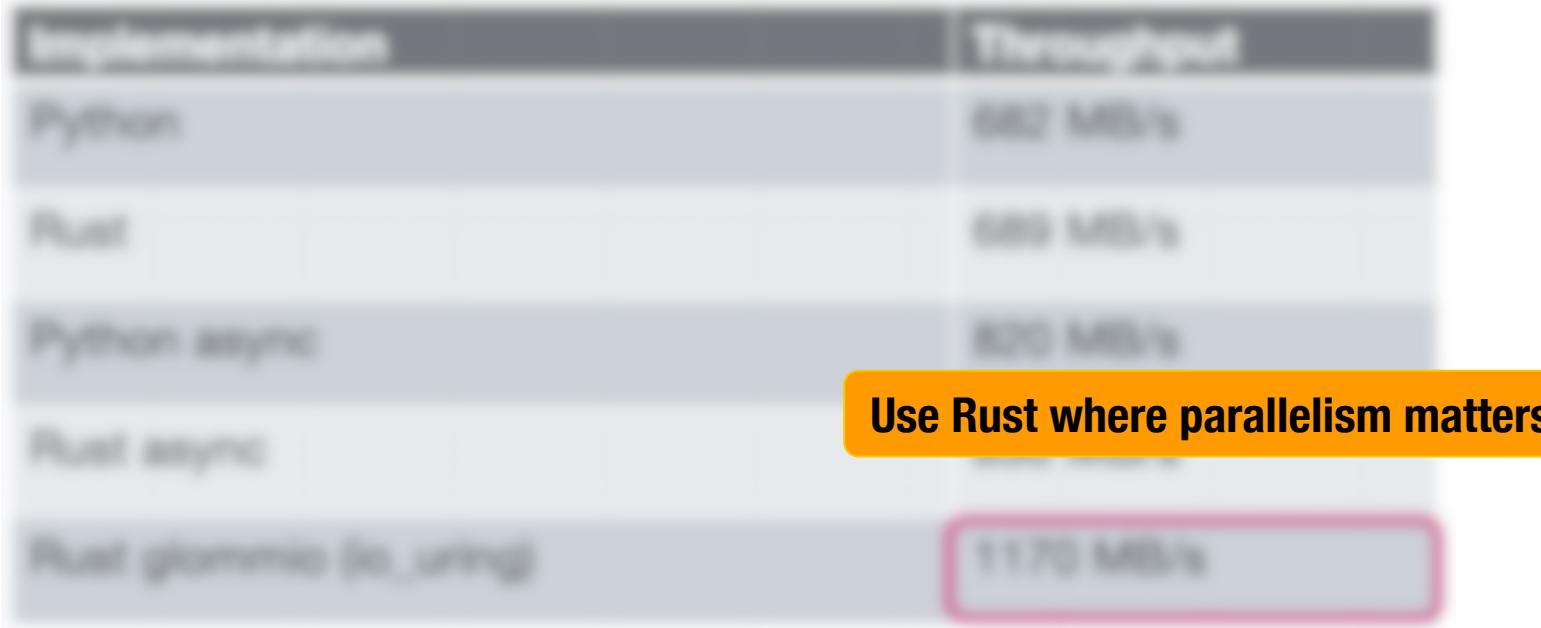
4 – Data management overhead

Implementation	Throughput
Python	682 MB/s
Rust	689 MB/s
Python async	820 MB/s
Rust async	830 MB/s

4 – Data management overhead

Implementation	Throughput
Python	682 MB/s
Rust	689 MB/s
Python async	820 MB/s
Rust async	830 MB/s
Rust glommio (io_uring)	1170 MB/s

4 – Data management overhead



**ONE DOES NOT SIMPLY
DEPLOY TO PRODUCTION**

5 – From PoC to Production

Future proof your APIs by focussing on how they might break

ONE DOES NOT SIMPLY
DEPLOY TO PRODUCTION

5 – From PoC to Production

Future proof your APIs by focussing on how they might break

Integration tests, integration tests and more integration tests

5 – From PoC to Production

Future proof your APIs by focussing on how they might break

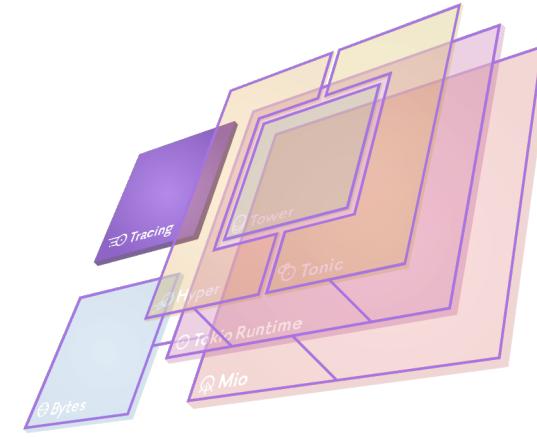
Integration tests, integration tests and more integration tests

Test as close to production as possible

6 – Observability



Structured, application-level diagnostics

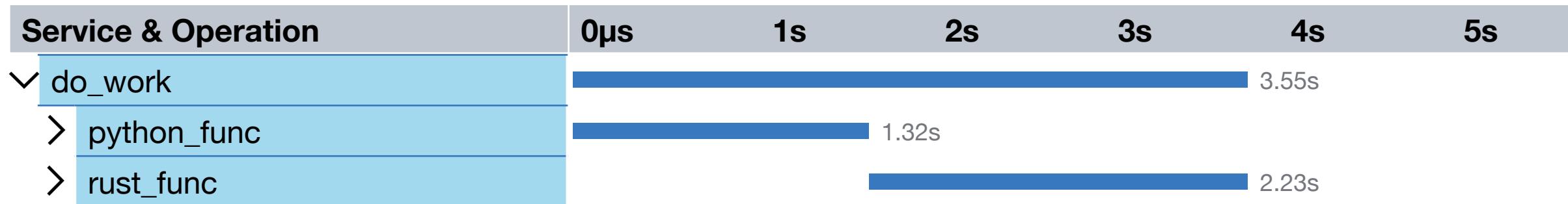


6 – Observability

```
6     @tracer.start_as_current_span("do_work")
7     def do_work():
8         python_func()
9         rust_func()
```

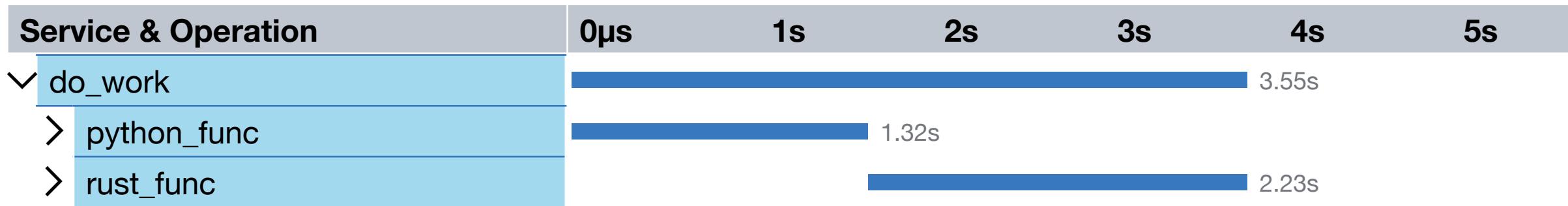
6 – Observability

```
6     @tracer.start_as_current_span("do_work")
7     def do_work():
8         python_func()
9         rust_func()
```



6 – Observability

```
6     @tracer.start_as_current_span("do_work")
7     def do_work():
8         python_func()
9         rust_func(get_current_span().get_span_context().span_id)
```



6 – Observability

```
32 #[pyfunction]
33 #[pyo3(name = "match_single_threaded")]
34 #[instrument(skip(py))]
35 pub fn main(py: Python, tasks: Vec<Task>, offers: Vec<Offer>) -> HashMap<String, String> {
```

6 – Observability

```
32 #[pyfunction]
33 #[pyo3(name = "match_single_threaded")]
34 #[instrument(skip(py))]
35 pub fn main(py: Python, tasks: Vec<Task>, offers: Vec<Offer>) -> HashMap<String, String> {
36     set_python_span_as_parent(py);
37     py.allow_threads(move || match_tasks_to_offers(tasks, offers))
38 }
```

6 – Observability

```
120  ✓ pub fn set_python_span_as_parent(py: Python) {  
121      let span: &Span = &tracing::Span::current();  
122      ✓ if span.is_disabled() {  
123          | return;  
124      }  
125  
126      let span_context: Result<Option<SpanContext>, ...> = get_python_opentelemetry_span_context(py);  
127  
128      ✓ if let Ok(Some(span_context: SpanContext)) = span_context {  
129          | let context: Context = opentelemetry::Context::new().with_remote_span_context(span_context);  
130          | span.set_parent(cx: context);  
131      }  
132  }  
133
```

6 – Observability

```
120  ✓ pub fn set_python_span_as_parent(py: Python) {
121      let span: &Span = &tracing::Span::current();
122  ✓     if span.is_disabled() {
123         return;
124     }
125
126     let span_context: Result<Option<SpanContext>, ...> = get_python_opentelemetry_span_context(py);
127
128     ✓ if let Ok(Some(span_context: SpanContext)) = span_context {
129         let context: Context = opentelemetry::Context::new().with_remote_span_context(span_context);
130         span.set_parent(cx: context);
131     }
132 }
133
```

6 – Observability

```
63 pub fn get_python_opentelemetry_span_context(
64     py: Python,
65 ) -> PyResult<Option<opentelemetry::trace::SpanContext>> {
66     static OPENTELEMETRY_TRACE_IMPORT: OnceCell<Py<PyModule>> = OnceCell::new();
67
68     let result: Result<&Py<PyModule>, PyErr> = OPENTELEMETRY_TRACE_IMPORT OnceCell<Py<P...
69         .get_or_try_init(|| py.import(name: "opentelemetry.trace").map(op: Into::into));
70     let module: &Bound<'_, PyModule> = match result { ...
71     };
72
73
74     // opentelemetry always returns a context, but maybe with an invalid span ID = 0
75     let python_span_context: Bound<'_, PyAny> = module &Bound<'_, PyModule>
76         .call_method0(name: intern!(py, "get_current_span"))? Bound<'_, PyAny>
77         .call_method0(name: intern!(py, "get_span_context"))?;
78
79     let trace_id: u128 = python_span_context Bound<'_, PyAny>
80         .getattr(attr_name: intern!(py, "trace_id"))? Bound<'_, PyAny>
81         .extract::<u128>()?;
82
83     if trace_id == 0 {
84         return Ok(None);
85     }
86 }
```

6 – Observability

```
63 pub fn get_python_opentelemetry_span_context(
64     py: Python,
65 ) -> PyResult<Option<opentelemetry::trace::SpanContext>> {
66     static OPENTELEMETRY_TRACE_IMPORT: OnceCell<Py<PyModule>> = OnceCell::new();
67
68     let result: Result<&Py<PyModule>, PyErr> = OPENTELEMETRY_TRACE_IMPORT OnceCell<Py<P...
69         .get_or_try_init(|| py.import(name: "opentelemetry.trace").map(op: Into::into));
70 >     let module: &Bound<'_, PyModule> = match result { ...
71     };
72
73
74 // opentelemetry always returns a context, but maybe with an invalid span ID = 0
75 let python_span_context: Bound<'_, PyAny> = module &Bound<'_, PyModule>
76     .call_method0(name: intern!(py, "get_current_span"))? Bound<'_, PyAny>
77     .call_method0(name: intern!(py, "get_span_context"))?;
78
79
80 let trace_id: u128 = python_span_context Bound<'_, PyAny>
81     .getattr(attr_name: intern!(py, "trace_id"))? Bound<'_, PyAny>
82     .extract::<u128>()?;
83
84
85 if trace_id == 0 {
86     return Ok(None);
87 }
88
89 }
```

6 – Observability

```
63 pub fn get_python_opentelemetry_span_context(
64     py: Python,
65 ) -> PyResult<Option<opentelemetry::trace::SpanContext>> {
66     static OPENTELEMETRY_TRACE_IMPORT: OnceCell<Py<PyModule>> = OnceCell::new();
67
68     let result: Result<&Py<PyModule>, PyErr> = OPENTELEMETRY_TRACE_IMPORT OnceCell<Py<P...
69         .get_or_try_init(|| py.import(name: "opentelemetry.trace").map(op: Into::into));
70 >     let module: &Bound<'_, PyModule> = match result { ...
71     };
72
73
74     // opentelemetry always returns a context, but maybe with an invalid span ID = 0
75     let python_span_context: Bound<'_, PyAny> = module &Bound<'_, PyModule>
76         .call_method0(name: intern!(py, "get_current_span"))? Bound<'_, PyAny>
77         .call_method0(name: intern!(py, "get_span_context"))?;
78
79     let trace_id: u128 = python_span_context Bound<'_, PyAny>
80         .getattr(attr_name: intern!(py, "trace_id"))? Bound<'_, PyAny>
81         .extract::<u128>()?;
82
83
84     if trace_id == 0 {
85         return Ok(None);
86     }
87 }
```

6 – Observability

```
88     if trace_id == 0 {
89         return Ok(None);
90     }
99
100    // create context
101    let span_context: SpanContext = opentelemetry::trace::SpanContext::new(
102        opentelemetry::trace::TraceId::from(trace_id),
103        opentelemetry::trace::SpanId::from(span_id),
104        opentelemetry::trace::TraceFlags::new(trace_flags),
105        is_remote: true, ←
106        trace_state: opentelemetry::trace::TraceState::NONE,
107    );
108
109    // Ensure span context is valid
110    > if !span_context.is_valid() { ...
115
116    Ok(Some(span_context))
117 } fn get_python_opentelemetry_span_context
```

6 – Observability

```
120  ✓ pub fn set_python_span_as_parent(py: Python) {  
121      let span: &Span = &tracing::Span::current();  
122  ✓     if span.is_disabled() {  
123         return;  
124     }  
125  
126     let span_context: Result<Option<SpanContext>, ...> = get_python_opentelemetry_span_context(py);  
127  
128  ✓     if let Ok(Some(span_context: SpanContext)) = span_context {  
129         let context: Context = opentelemetry::Context::new().with_remote_span_context(span_context);  
130         span.set_parent(cx: context);  
131     }  
132 }  
133
```

7 – From PoC to Production

Build Rust using the release profile

Version compatibility: PyO3 – rustc – python

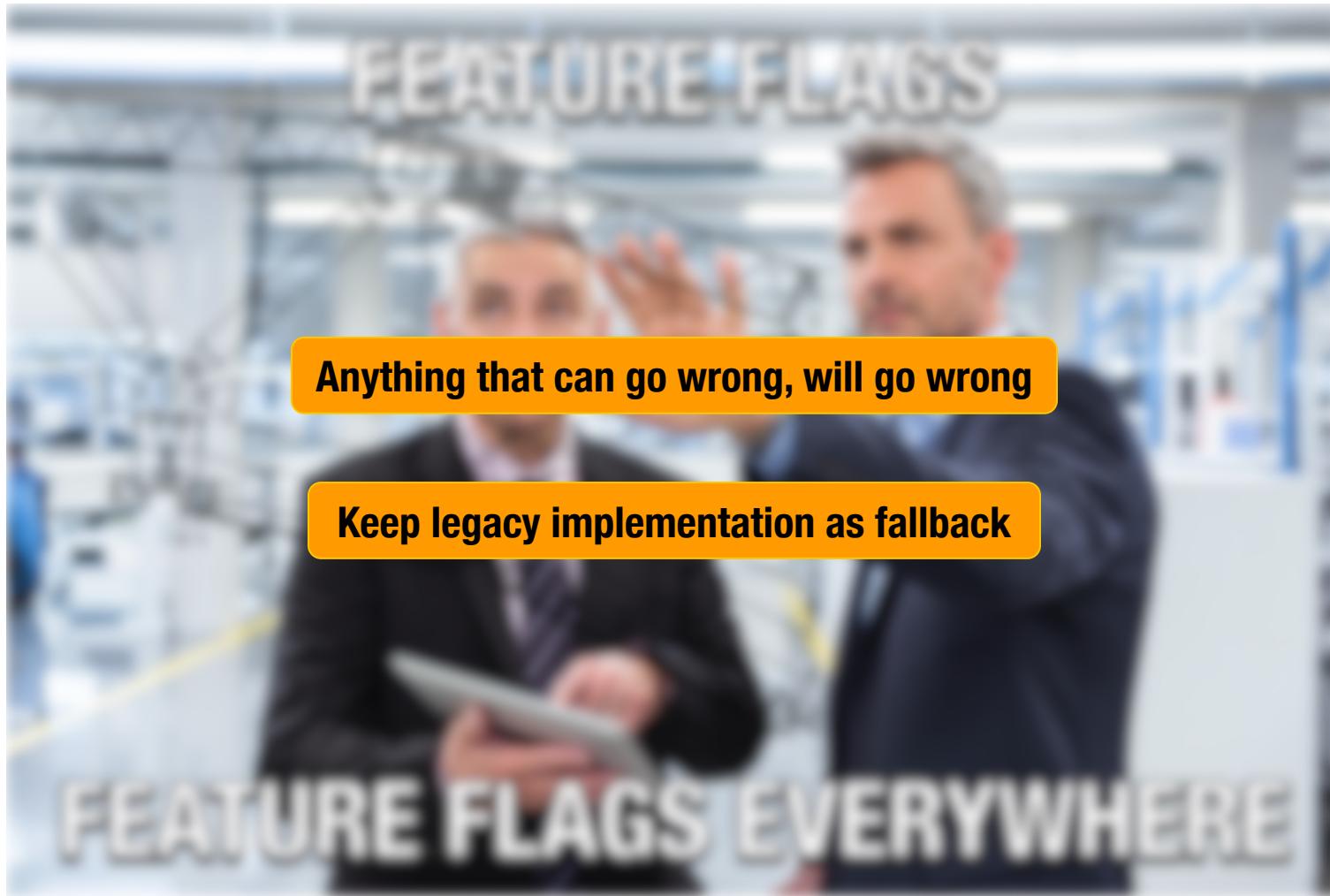
7 – From PoC to Production



7 – From PoC to Production



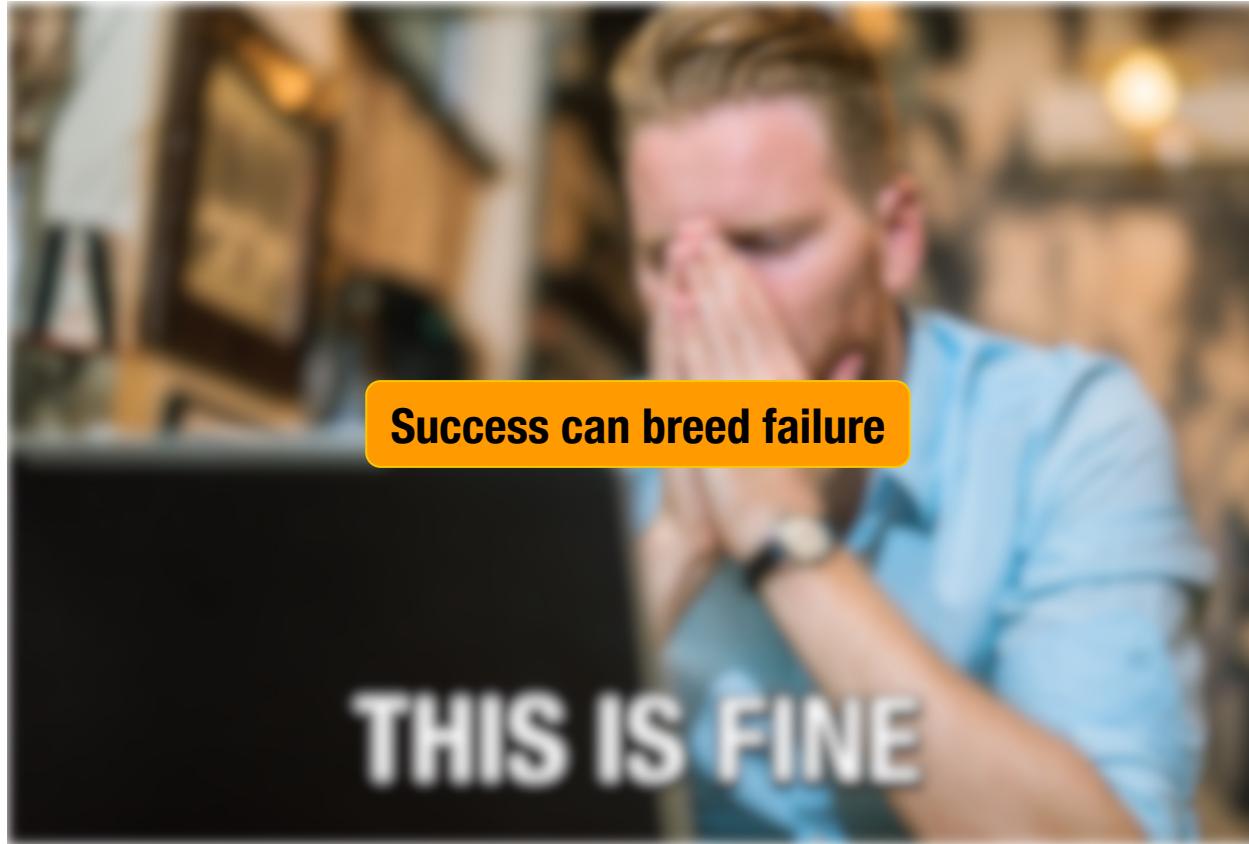
7 – From PoC to Production



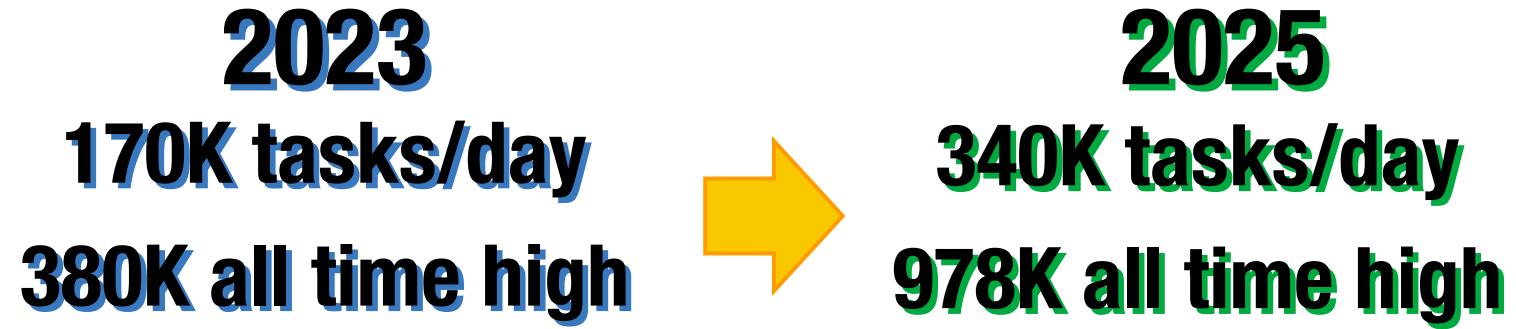
7 – From PoC to Production



7 – From PoC to Production



8 – The End



8 – The End

