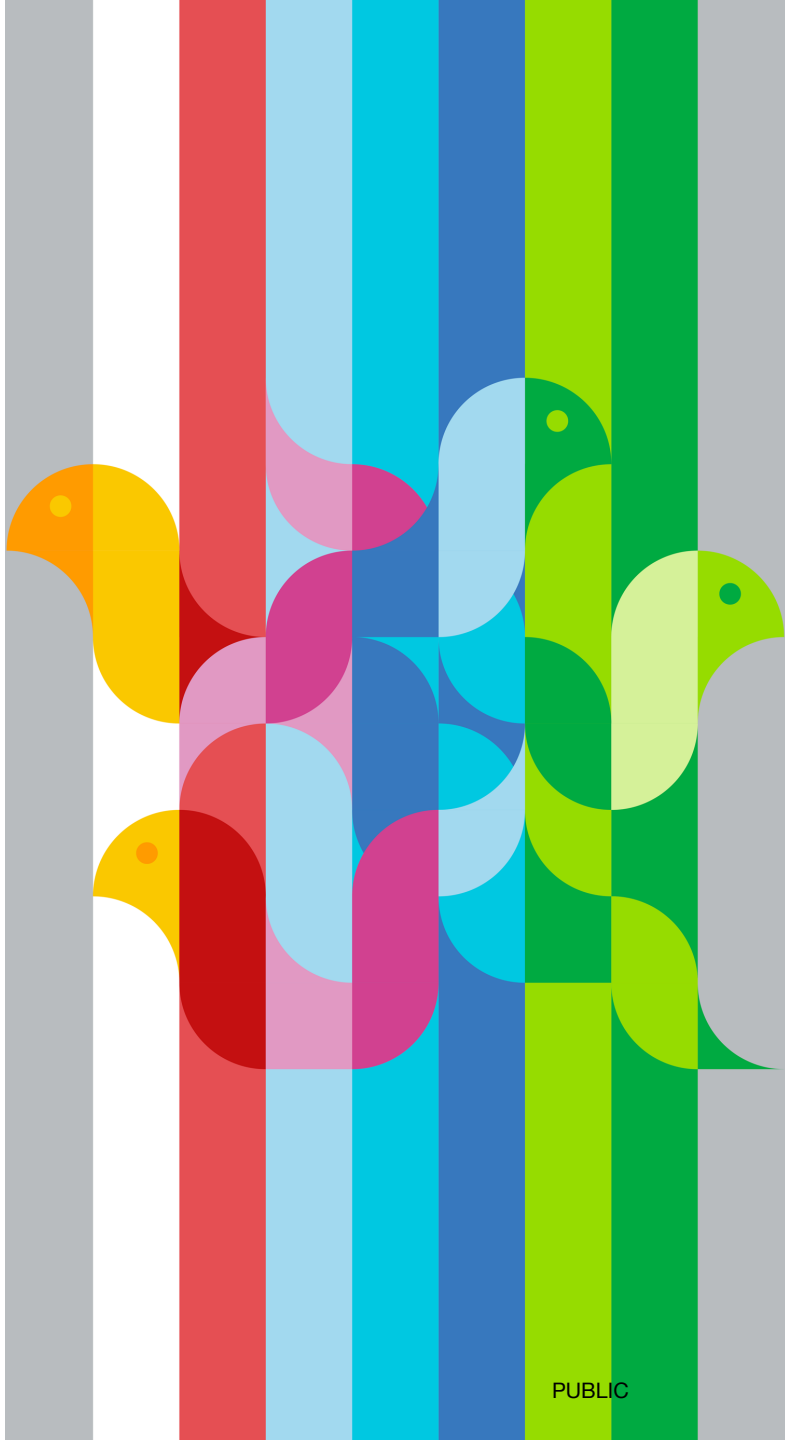
PyConDE
PyData
Darmstadt
April 2025

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

Max Höhl
SAP SE

/whme/PyConDE-2025



/whme/PyConDE-2025

PUBLIC

> **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

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

/whme/PyConDE-2025

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

/whme/PyConDE-2025

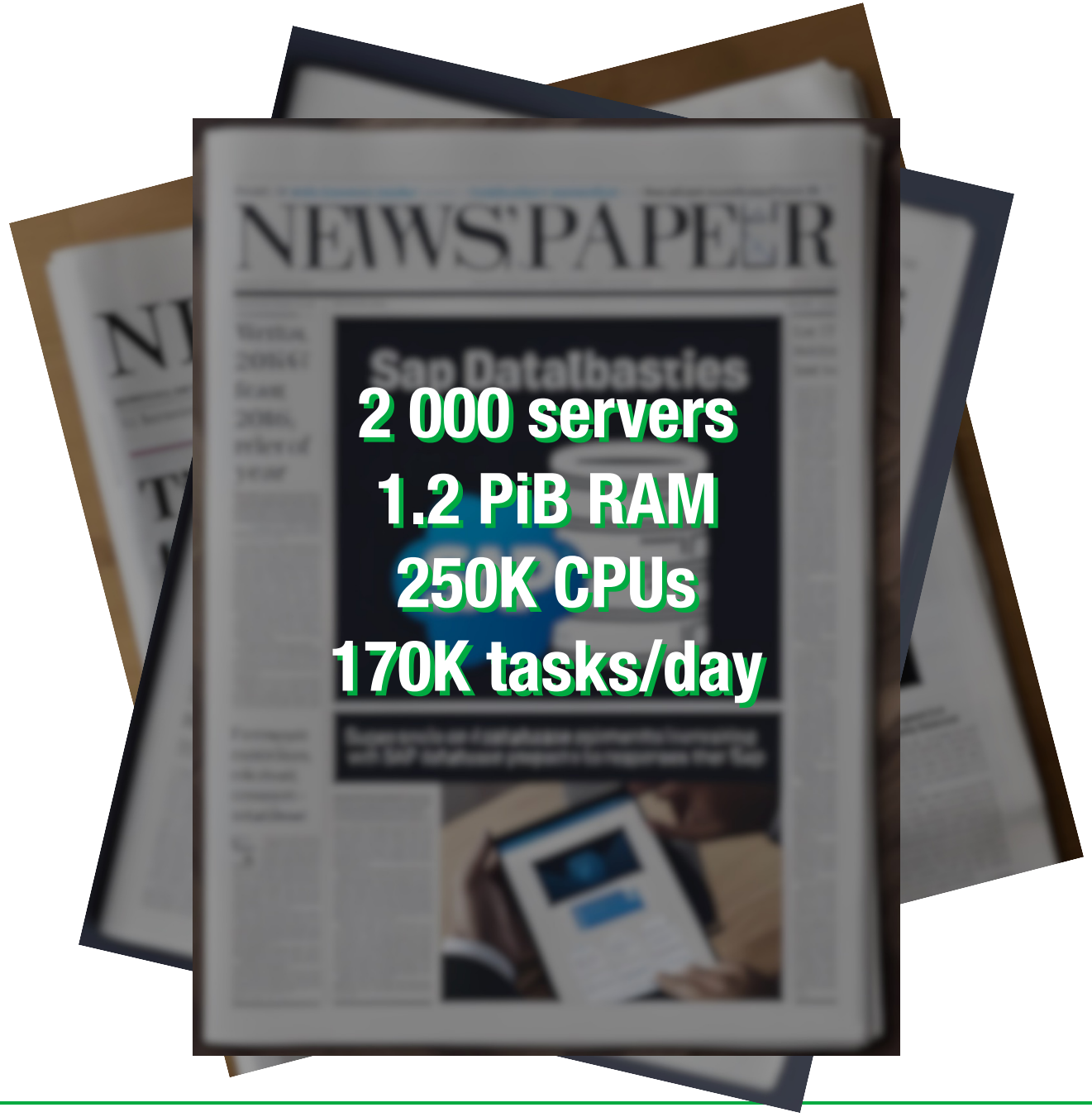
- How to identify parts suitable for Rust
- How to migrate to Rust and deploy to production
- Lessons learned

/whme/PyConDE-2025

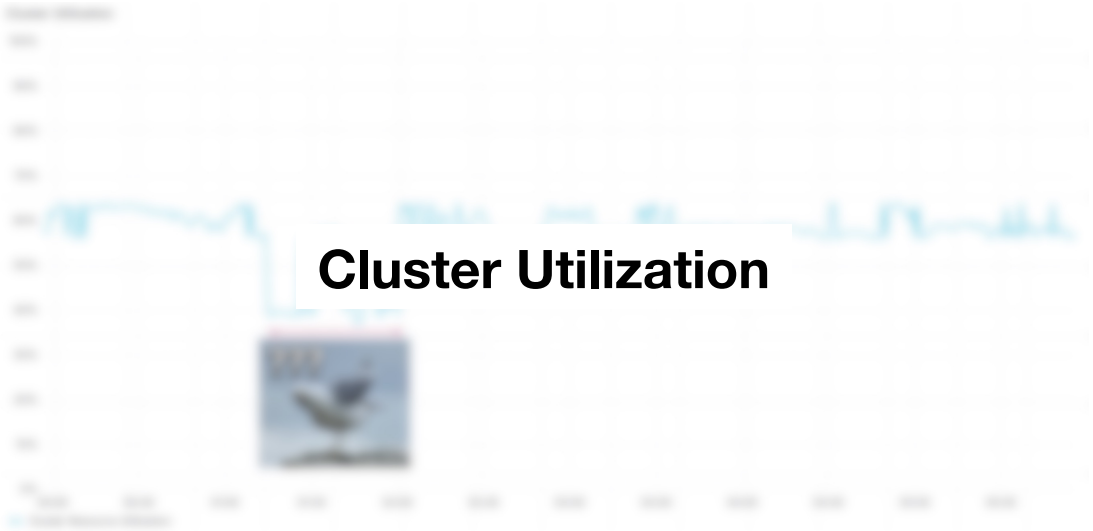

NEWS FSII6
The fellow Eour 10 purisst
6.16
Tianc garsevie havix buv With Drexit the vox
BREXIT
In brexveling brexit is a Brexit and rarefs worle


Beginıng of 2016
Timeır Smartphones with up
tourdıoms and prue pow
Smartphone dowvards arrow
retier up at tlıe PC
PC


NEWS'PAPER
Sap Databasties
SAP


NEWSPAPER
2 000 servers
1.2 PiB RAM
250K CPUs
170K tasks/day

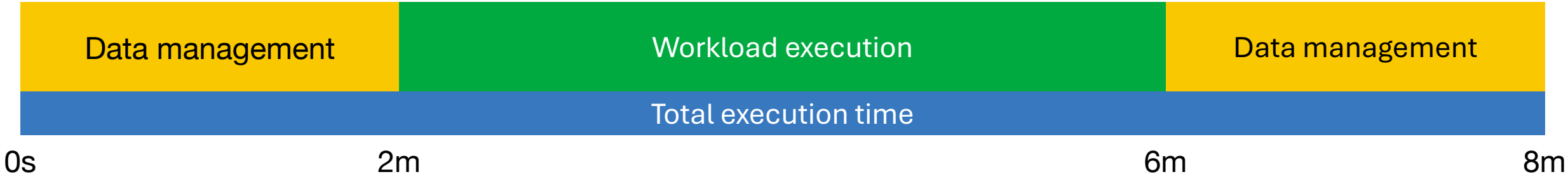
1 – Conflict



1 – Conflict


Cluster Utilization
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
00:00
00:30
01:00
01:30
02:00
02:30
03:00
03:30
04:00
04:30
05:00
05:30
???
Cluster Resource Utilization

1 – Conflict


Data management
Workload execution
Data management
Total execution time
0s
2m
6m
8m

2 – Cluster Utilization

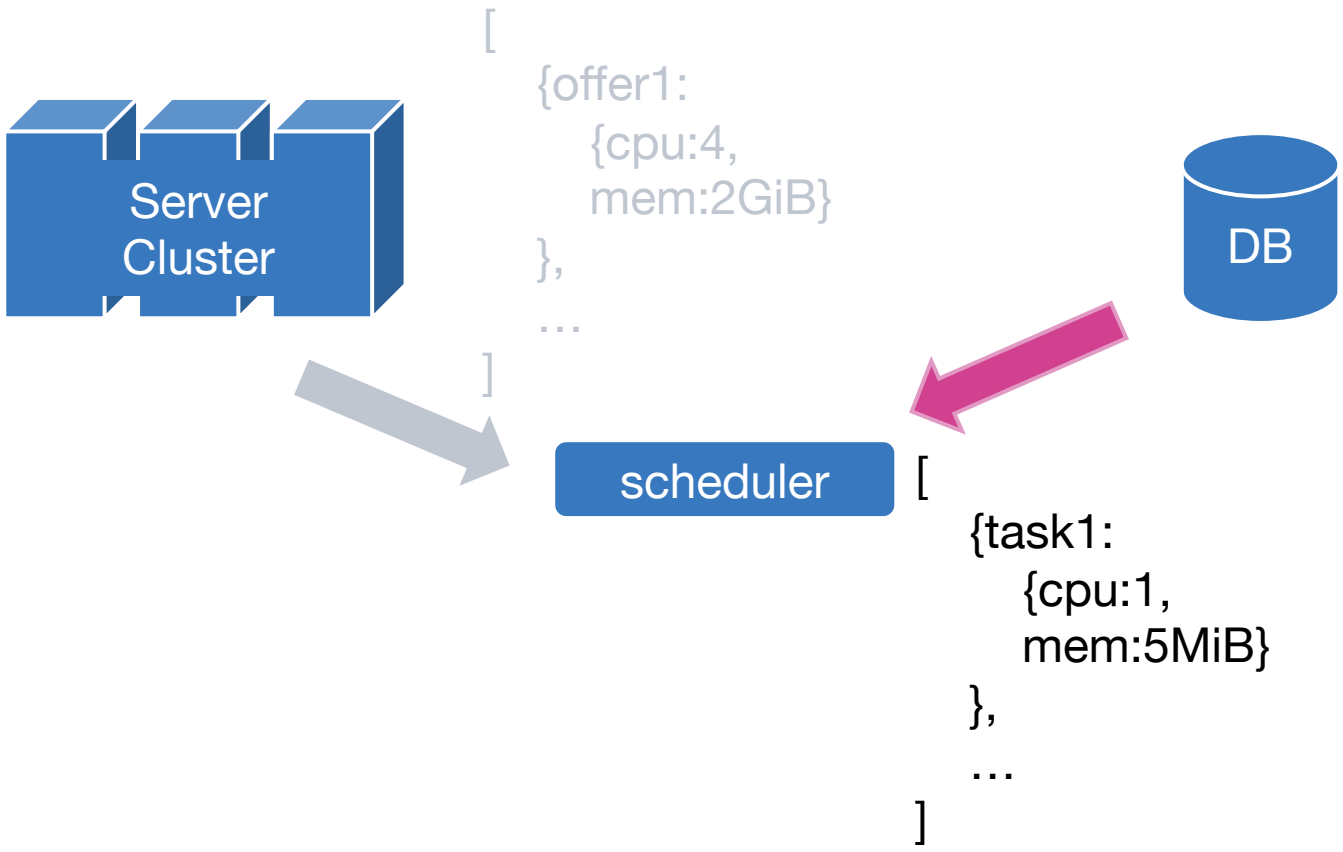
get_resource_offers_from_cluster


```
[
  {offer1:
    {cpu:4,
    mem:2GiB}
  },
  …
]
```

scheduler

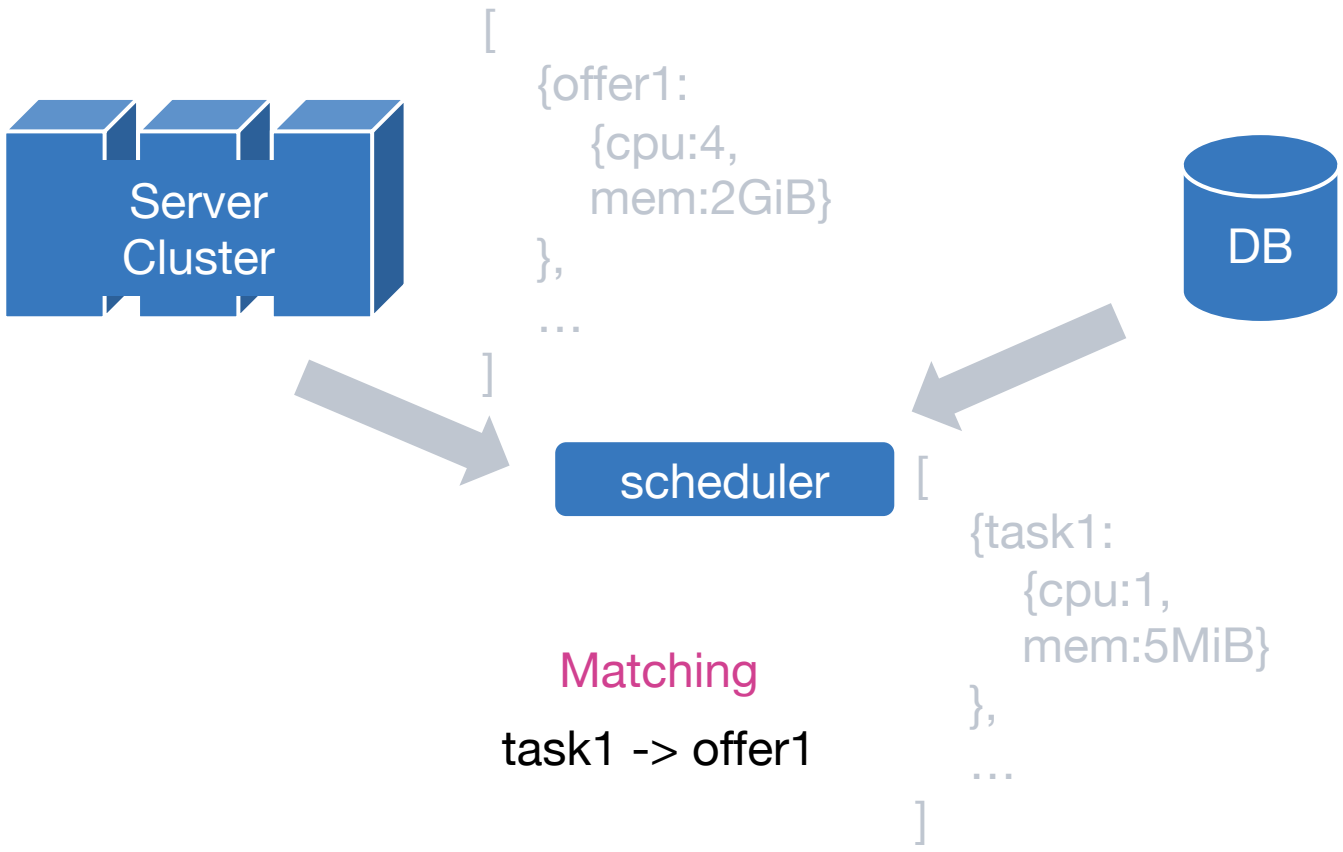
2 – Cluster Utilization

- get_resource_offers_from_cluster
- get_processable_tasks



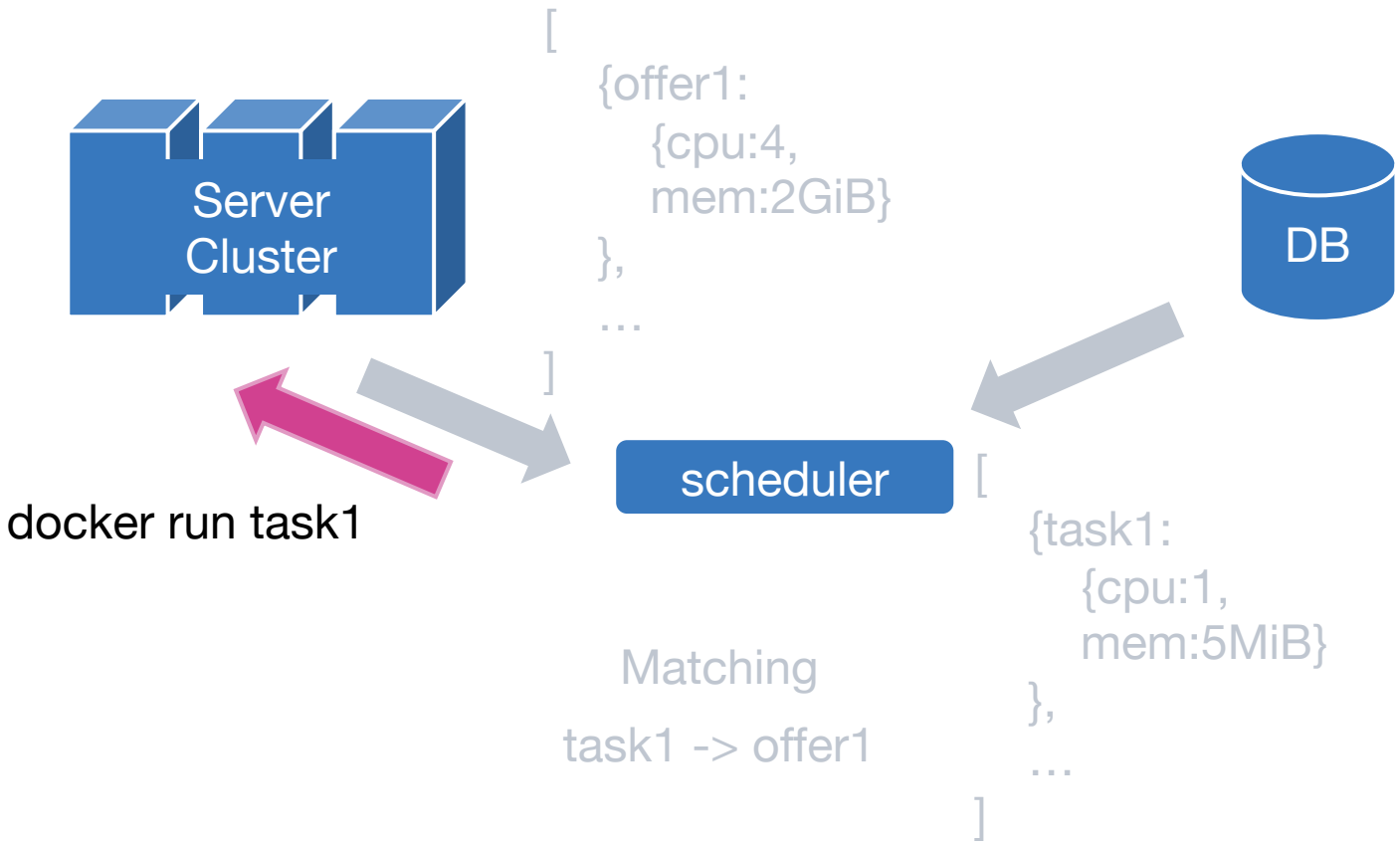
2 – Cluster Utilization

- get_resource_offers_from_cluster
- get_processable_tasks
- match_tasks_to_offers



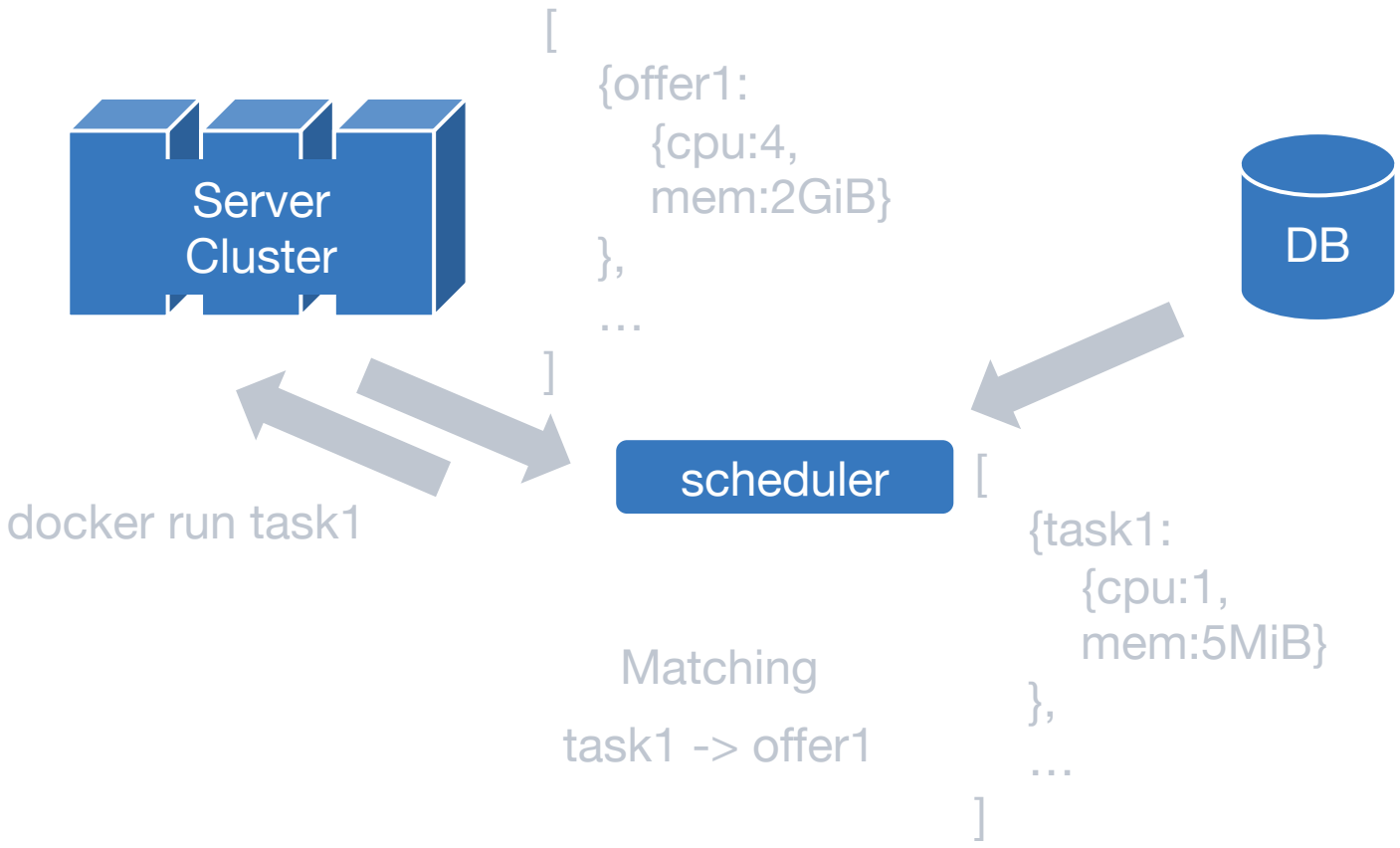
2 – Cluster Utilization

get_resource_offers_from_cluster
get_processable_tasks
match_tasks_to_offers
prepare_payloads

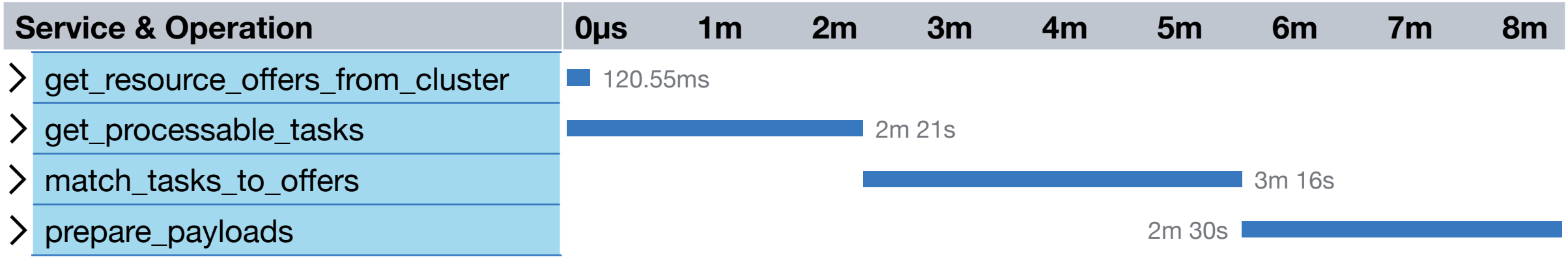


2 – Cluster Utilization

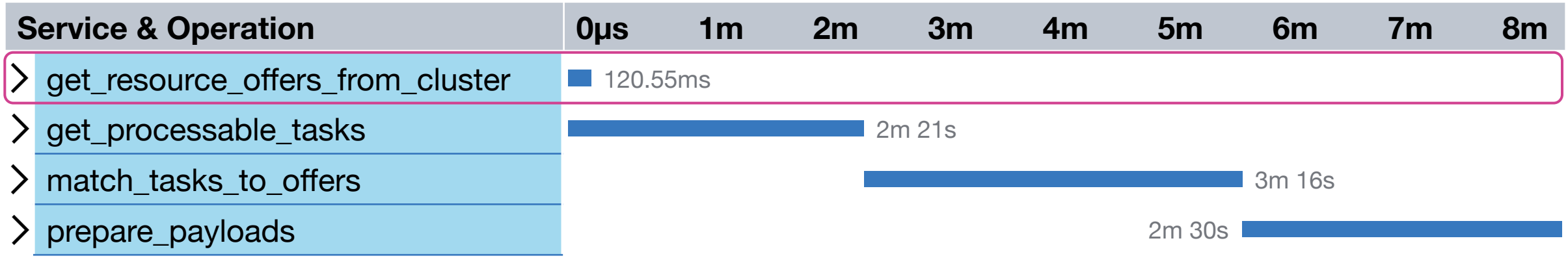
get_resource_offers_from_cluster
get_processable_tasks
match_tasks_to_offers
prepare_payloads



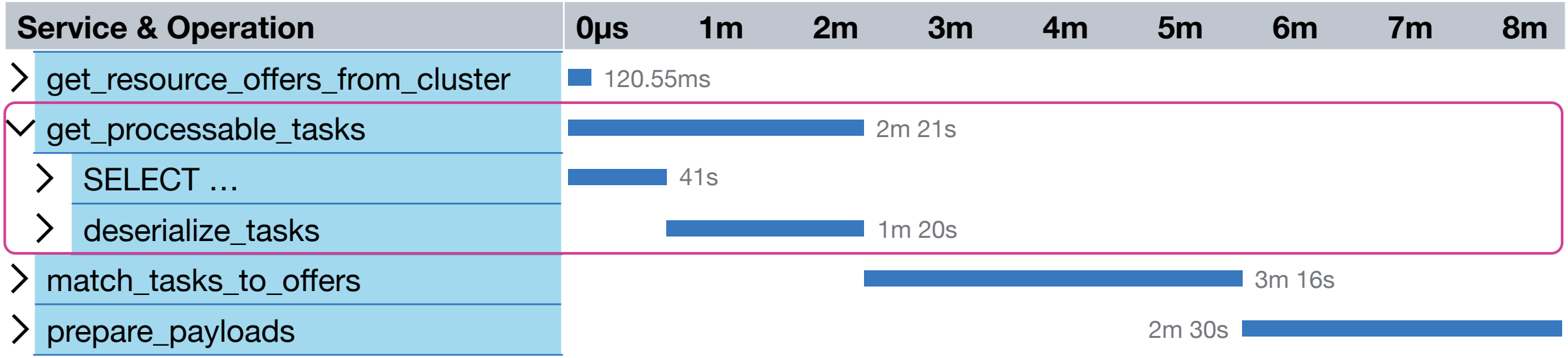
2 – Cluster Utilization

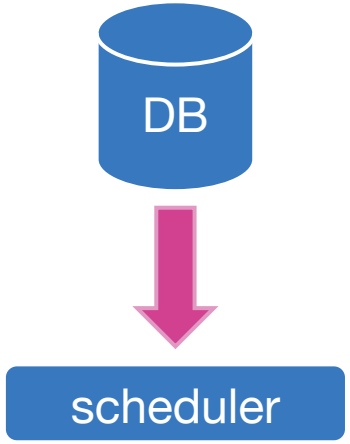


2 – Cluster Utilization



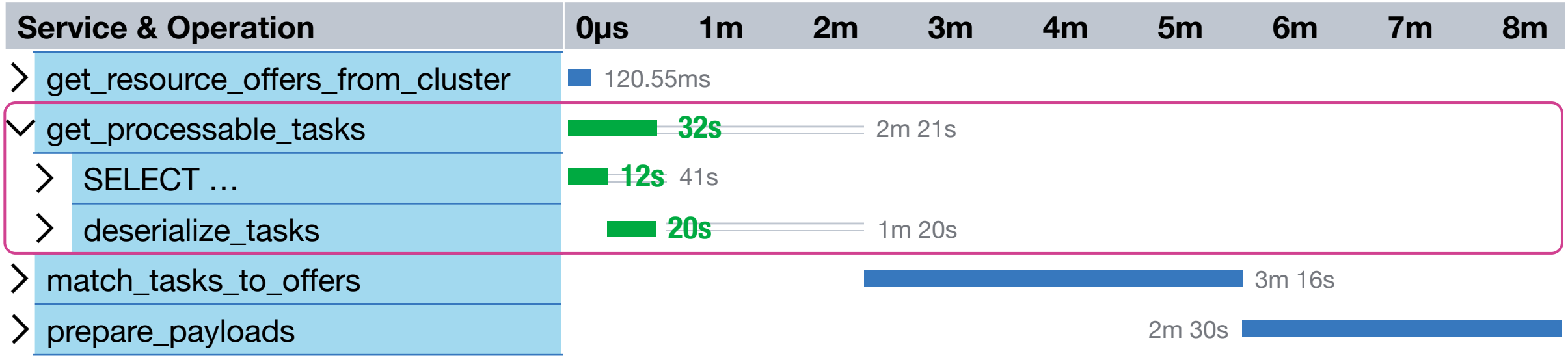
2 – Cluster Utilization

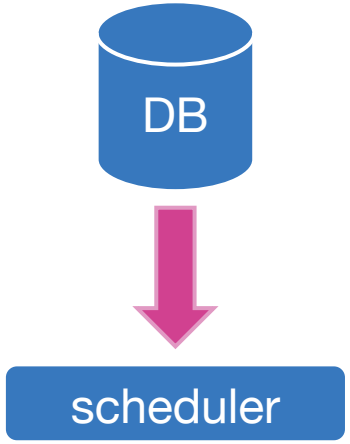





```
[
    {task3: {depends_on: [{task2: …}, …]}},
    …
]
```

2 – Cluster Utilization





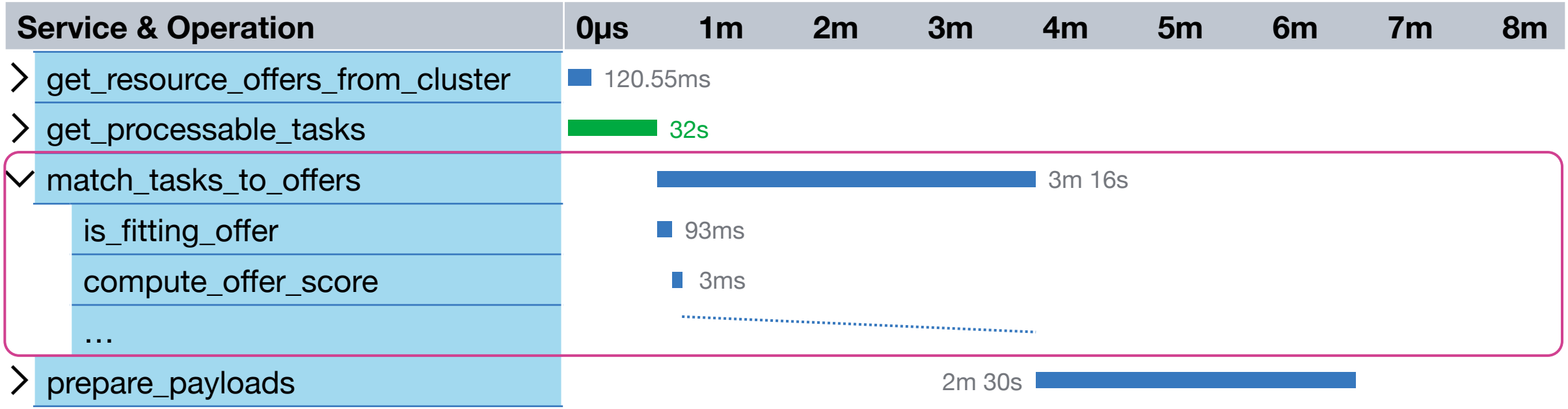

```
[
    {task3: {depends_on: [task2, …]}},
    …
]
```

2 – Cluster Utilization

~~Silver bullet~~

Focus on understanding and fixing the underlying root cause

2 – Cluster Utilization



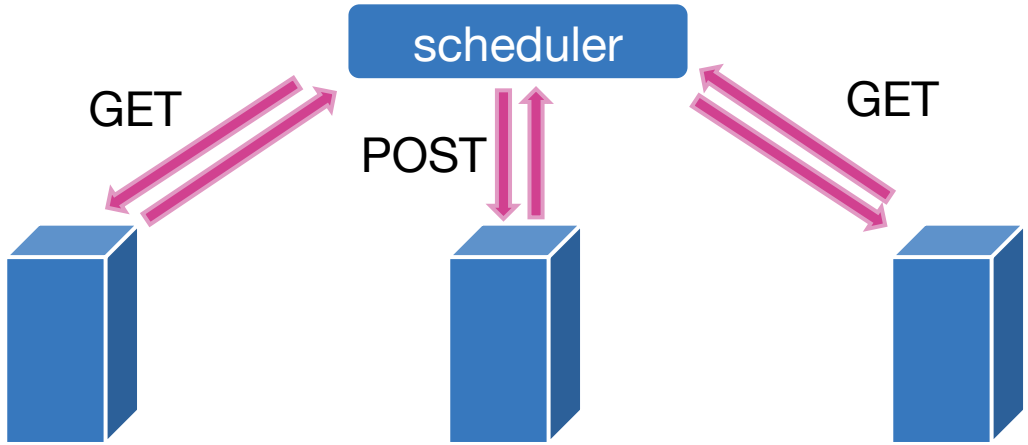
offer1: {cpu: 10, mem: 200GiB, …}
offer2: {cpu: 150, mem: 1.5TiB, …}
…

task1: {cpu: 5, mem: 100GiB, …}
task2: {cpu: 145, mem: 200GiB, …}
…

offer1: []
offer2: [task1, task2]

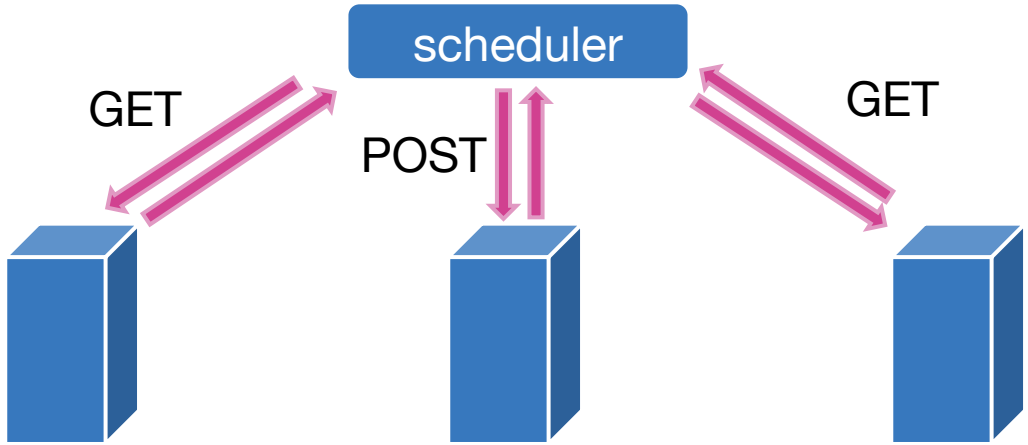
2 – Cluster Utilization

Service & Operation	0µs	1m	2m	3m	4m	5m	6m	7m	8m
> get_resource_offers_from_cluster	120.55ms								
> get_processable_tasks	32s								
> match_tasks_to_offers					3m 16s				
v prepare_payloads					2m 30s				
GET …					913.71ms				
POST …					932.81ms				
…									



2 – Cluster Utilization

Service & Operation	0µs	1m	2m	3m	4m	5m	6m	7m	8m
> get_resource_offers_from_cluster	120.55ms								
> get_processable_tasks	32s								
> match_tasks_to_offers		3m 16s							
v prepare_payloads					2m 30s	23s			
GET …					913.71ms				
POST …					932.81ms				
…									



2 – Cluster Utilization

Use existing toolset to its fullest potential

3 – Offer Matching

```python
def match_tasks_to_offers(
    tasks: list[Task], offers: list[Offer]
) -> dict[OfferName, TaskName]:
    """Returns a mapping from offer name to task name where the key is the name
    of the bestfit offer for the task.
    """
    result = {}
    for task in tasks:
        best_index_score: tuple[OfferIndex | None, float] = (None, 31.0)
        for index, offer in enumerate(offers):
            if is_fitting_offer(task, offer):
                score = compute_score(task, offer)
                if score < best_index_score[1]:
                    best_index_score = (index, score)
        if (best_index := best_index_score[0]) is not None:
            result[swap_remove(offers, best_index).name] = task.name

    return result
```

3 – Offer Matching

```python
def match_tasks_to_offers(
    tasks: list[Task], offers: list[Offer]
) -> dict[OfferName, TaskName]:
    """Returns a mapping from offer name to task name where the key is the name
    of the bestfit offer for the task.
    """
    result = {}
    for task in tasks:
        best_index_score: tuple[OfferIndex | None, float] = (None, 31.0)
        for index, offer in enumerate(offers):
            if is_fitting_offer(task, offer):
                score = compute_score(task, offer)
                if score < best_index_score[1]:
                    best_index_score = (index, score)
        if (best_index := best_index_score[0]) is not None:
            result[swap_remove(offers, best_index).name] = task.name

    return result
```

3 – Offer Matching

```python
def match_tasks_to_offers(
    tasks: list[Task], offers: list[Offer]
) -> dict[OfferName, TaskName]:
    """Returns a mapping from offer name to task name where the key is the name
    of the bestfit offer for the task.
    """
    result = {}
    for task in tasks:
        best_index_score: tuple[OfferIndex | None, float] = (None, 31.0)
        for index, offer in enumerate(offers):
            if is_fitting_offer(task, offer):
                score = compute_score(task, offer)
                if score < best_index_score[1]:
                    best_index_score = (index, score)
        if (best_index := best_index_score[0]) is not None:
            result[swap_remove(offers, best_index).name] = task.name

    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

```python
def match_tasks_to_offers(
    pool_class: type[Pool], tasks: list[Task], offers: list[Offer]
) -> dict[OfferName, TaskName]:
    """Match the given tasks to the given offers"""
    pool = pool_class(os.cpu_count())
    result = {}
    for task in tasks:
        best_index_score: tuple[OfferIndex | None, float] = (None, 31.0)
        for index, score in enumerate(
            pool.starmap(get_offer_score, [(task, offer) for offer in offers])
        ):
            if score is not None and score < best_index_score[1]:
                best_index_score = (index, score)
        if best_index_score[0] is None:
            continue
        result[(swap_remove(offers, best_index_score[0]).name)] = task.name
    pool.close()
    pool.join()
    return result
```

3 – Offer Matching

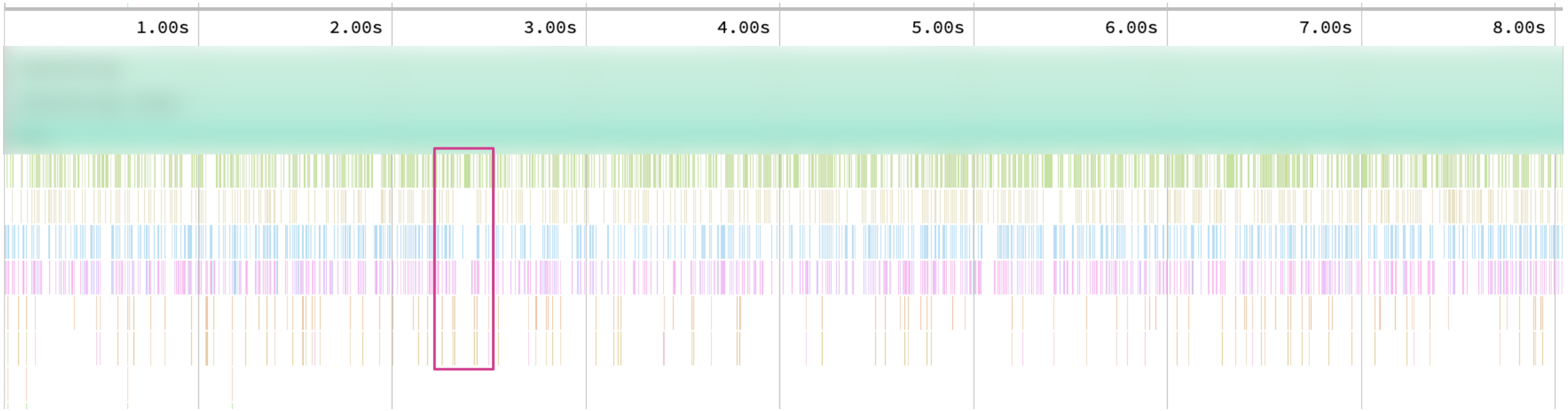
```python
def match_tasks_to_offers(
    pool_class: type[Pool], tasks: list[Task], offers: list[Offer]
) -> dict[OfferName, TaskName]:
    """Match the given tasks to the given offers"""
    pool = pool_class(os.cpu_count())
    result = {}
    for task in tasks:
        best_index_score: tuple[OfferIndex | None, float] = (None, 31.0)
        for index, score in enumerate(
            pool.starmap(get_offer_score, [(task, offer) for offer in offers])
        ):
            if score is not None and score < best_index_score[1]:
                best_index_score = (index, score)
        if best_index_score[0] is None:
            continue
        result[(swap_remove(offers, best_index_score[0]).name)] = task.name
    pool.close()
    pool.join()
    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

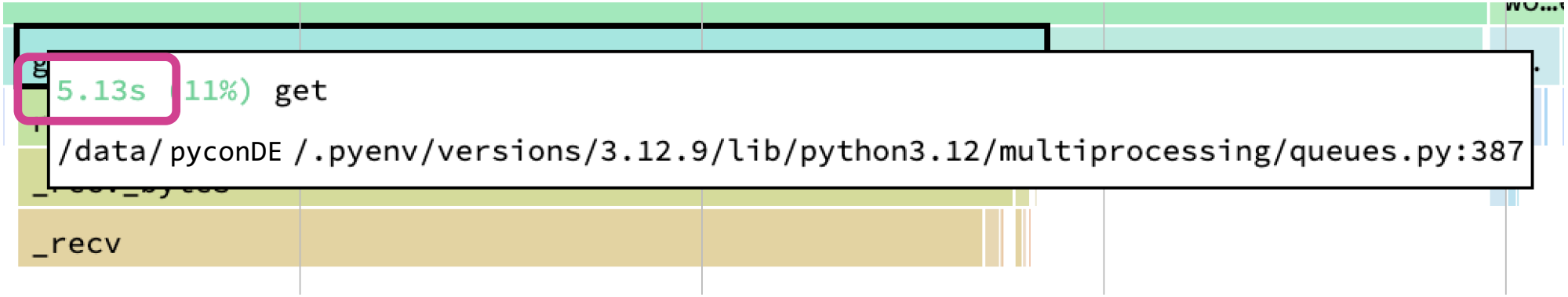


3 – Offer Matching

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

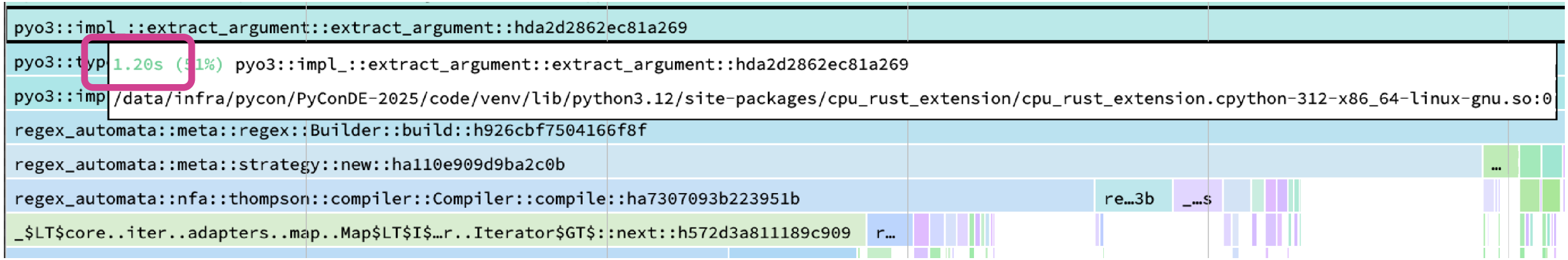

```rust
fn match_tasks_to_offers(tasks: Vec<Task>, mut offers: Vec<Offer>) -> HashMap<String, String> {
    let mut result = HashMap::<String, String>::with_capacity(offers.len());
    let mut best_index_score: (Option<usize>, f64);
    for task in tasks.iter() {
        best_index_score = (None, 31.0);
        for (index, offer) in offers.iter().enumerate() {
            if is_fitting_offer(task, offer) {
                let score = compute_score(task, offer);
                if score < best_index_score.1 {
                    best_index_score = (Some(index), score);
                }
            }
        }
        if let Some(best_index) = best_index_score.0 {
            result.insert(offers.swap_remove(best_index).name, task.name.clone());
        }
    }
    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
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

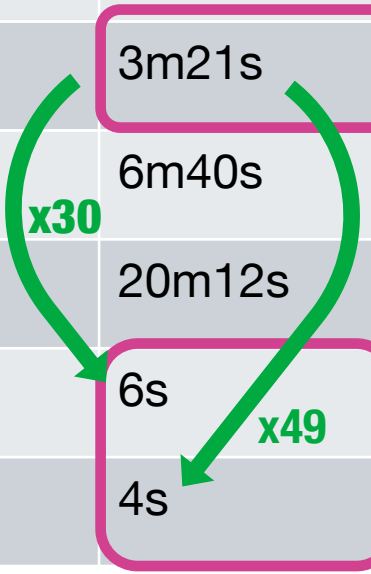
Implementation	Total time		is_fitting_offer time	compute_score time
	10K Tasks 2K Offers	20K Tasks 20K Offers		
Production benchmark	-	3m16s	93ms	3ms
Python single threaded	2s		83ms	2ms
Python multi threaded	27s		495ms	29ms
Python multi processing	70s		50ms	2ms
Rust single threaded	2s		20ms	150µs
Rust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

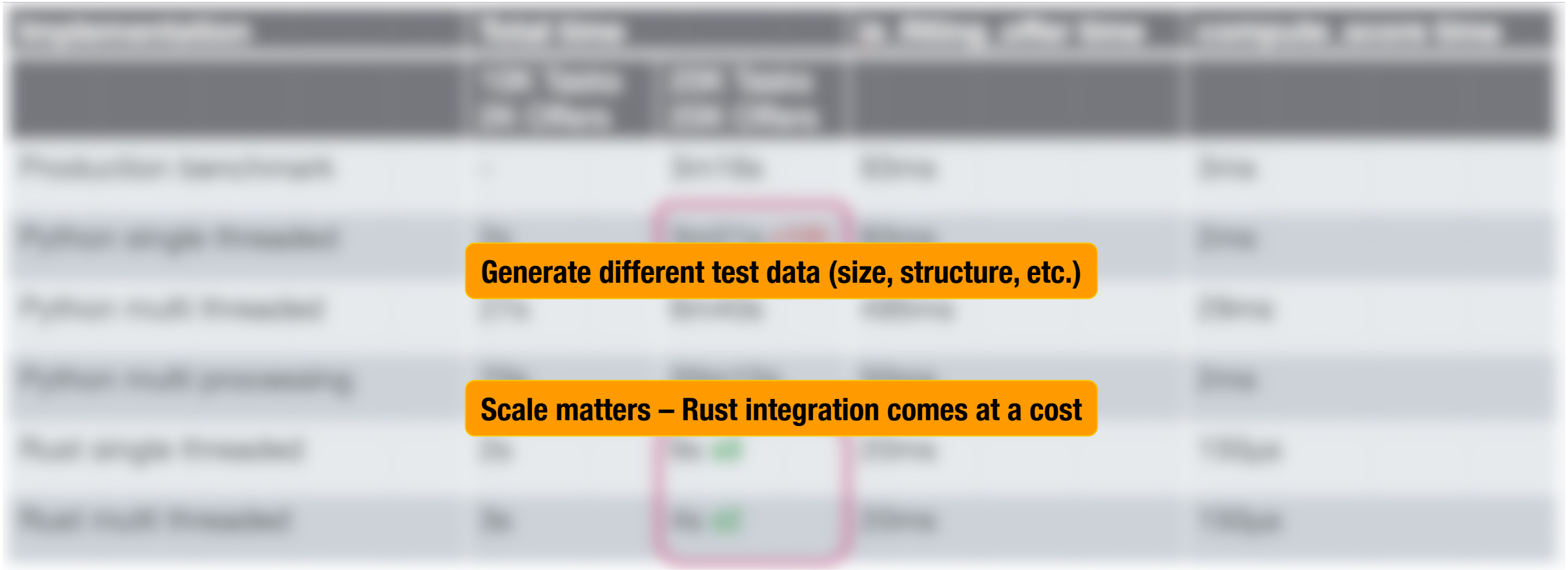
3 – Offer Matching

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	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	20ms	150µs
Rust multi threaded	3s	4s	20ms	150µs



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

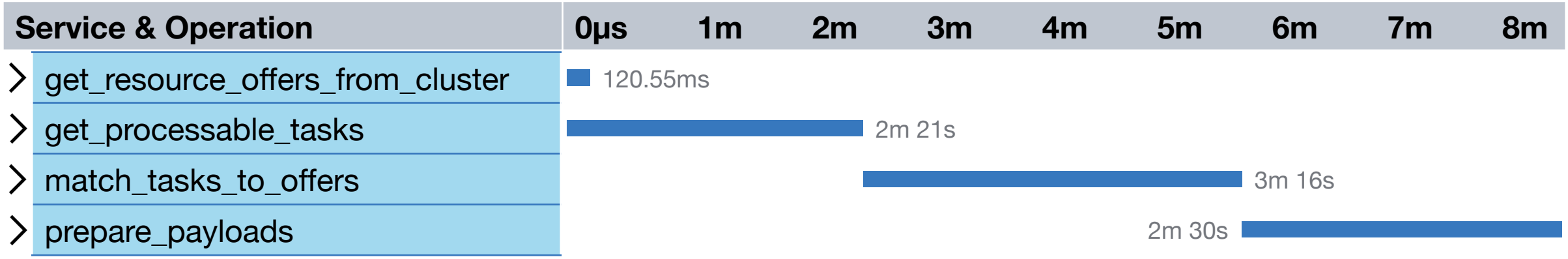
```
32    32 #[pyfunction]
33    33 #[pyo3(name = "match_single_threaded")]
34       pub fn main(tasks: Vec<Task>, offers: Vec<Offer>) -> HashMap<String, String> {
35           match_tasks_to_offers(tasks, offers)
      34 pub fn main(py: Python, tasks: Vec<Task>, offers: Vec<Offer>) -> HashMap<String,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

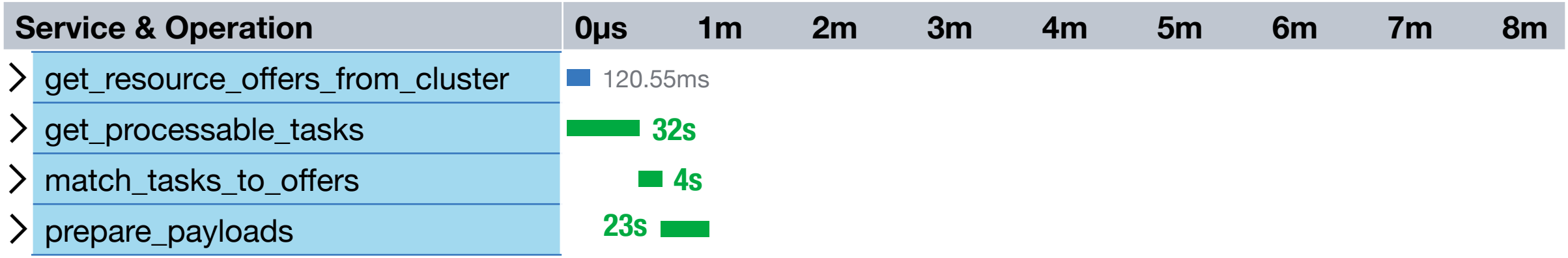
3 – Offer Matching

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

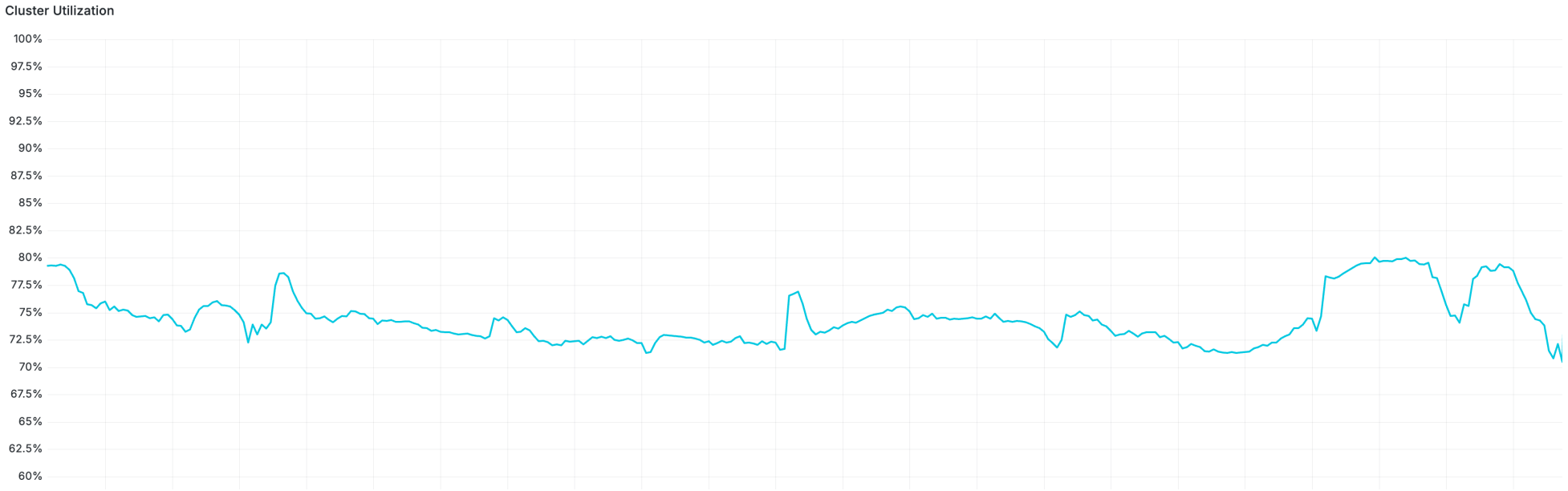
```
32  32  #[pyfunction]
33  33  #[pyo3(name = "match_single_threaded")]
34      pub fn main(tasks: Vec<Task>, offers: Vec<Offer>) -> HashMap<String, String> {
35          match_tasks_to_offers(tasks, offers)
    34  pub fn main(py: Python, tasks: Vec<Task>, offers: Vec<Offer>) -> HashMap<String, String>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

3 – Offer Matching



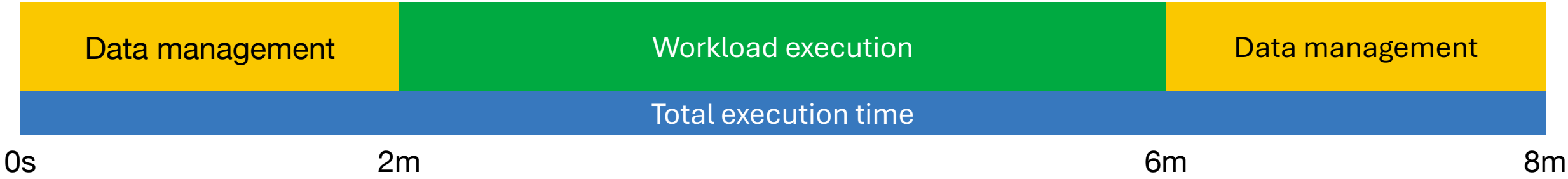


3 – Offer Matching



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

4 – Data management



4 – Data management

”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

”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

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

4 – Data management

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

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

4 – Data management

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

4 – Data management

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

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

Use Rust where parallelism matters

ONE DOES NOT SIMPLY DEPLOY TO PRODUCTION

5 – From PoC to Production

Future proof your APIs by focussing on how they might break

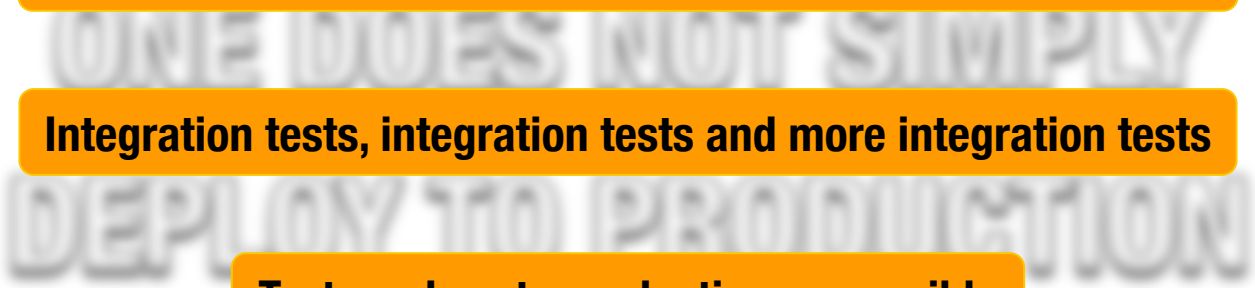
5 – From PoC to Production

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



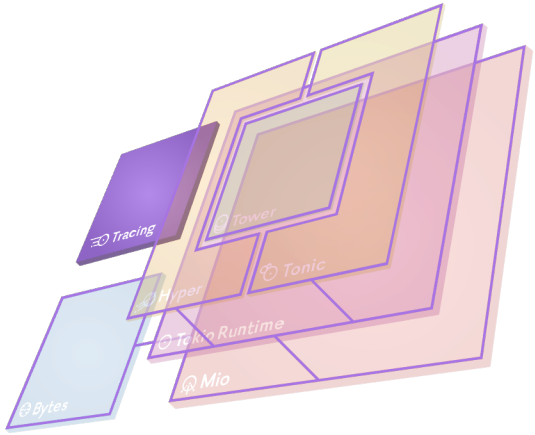
Integration tests, integration tests and more integration tests

Test as close to production as possible

6 – Observability

Tracing

Structured, application-level diagnostics



SENTRY

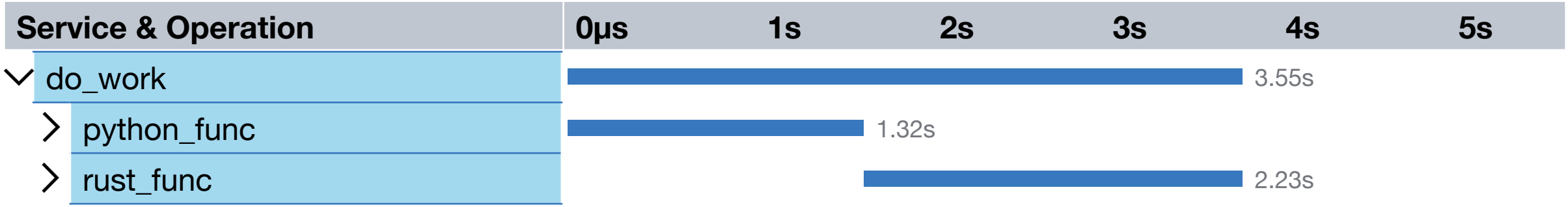
OpenTelemetry

6 – Observability


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

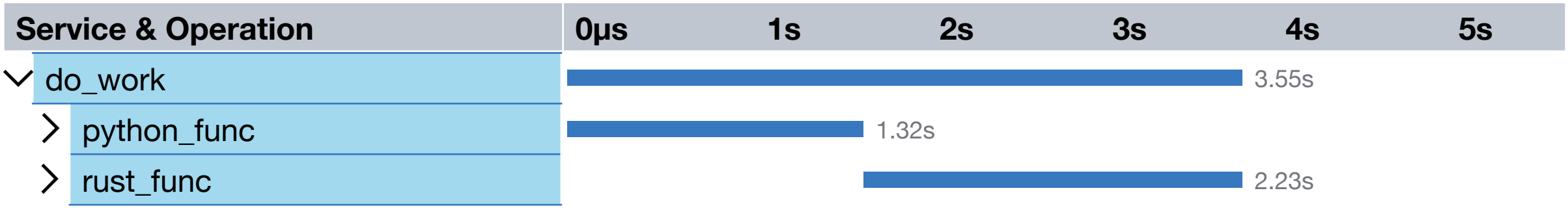
6 – Observability


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



6 – Observability


```python
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> {
36      set_python_span_as_parent(py);
37      py.allow_threads(move || match_tasks_to_offers(tasks, offers))
38  }
```

6 – Observability

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

6 – Observability

```rust
pub fn set_python_span_as_parent(py: Python) {
    let span: &Span = &tracing::Span::current();
    if span.is_disabled() {
        return;
    }

    let span_context: Result<Option<SpanContext>, …> = get_python_opentelemetry_span_context(py);

    if let Ok(Some(span_context: SpanContext)) = span_context {
        let context: Context = opentelemetry::Context::new().with_remote_span_context(span_context);
        span.set_parent(cx: context);
    }
}
```

6 – Observability

```rust
pub fn set_python_span_as_parent(py: Python) {
    let span: &Span = &tracing::Span::current();
    if span.is_disabled() {
        return;
    }

    let span_context: Result<Option<SpanContext>, …> = get_python_opentelemetry_span_context(py);

    if let Ok(Some(span_context: SpanContext)) = span_context {
        let context: Context = opentelemetry::Context::new().with_remote_span_context(span_context);
        span.set_parent(cx: context);
    }
}
```

6 – Observability

```rust
pub fn get_python_opentelemetry_span_context(
    py: Python,
) -> PyResult<Option<opentelemetry::trace::SpanContext>> {
    static OPENTELEMETRY_TRACE_IMPORT: OnceCell<Py<PyModule>> = OnceCell::new();

    let result: Result<&Py<PyModule>, PyErr> = OPENTELEMETRY_TRACE_IMPORT OnceCell<Py<P…
        .get_or_try_init(|| py.import(name: "opentelemetry.trace").map(op: Into::into));
    let module: &Bound<'_, PyModule> = match result {…
    };

    // opentelemetry always returns a context, but maybe with an invalid span ID = 0
    let python_span_context: Bound<'_, PyAny> = module &Bound<'_, PyModule>
        .call_method0(name: intern!(py, "get_current_span"))? Bound<'_, PyAny>
        .call_method0(name: intern!(py, "get_span_context"))?;

    let trace_id: u128 = python_span_context Bound<'_, PyAny>
        .getattr(attr_name: intern!(py, "trace_id"))? Bound<'_, PyAny>
        .extract::<u128>()?;

    if trace_id == 0 {
        return Ok(None);
    }
```

6 – Observability

```rust
pub fn get_python_opentelemetry_span_context(
    py: Python,
) -> PyResult<Option<opentelemetry::trace::SpanContext>> {
    static OPENTELEMETRY_TRACE_IMPORT: OnceCell<Py<PyModule>> = OnceCell::new();

    let result: Result<&Py<PyModule>, PyErr> = OPENTELEMETRY_TRACE_IMPORT OnceCell<Py<P…
        .get_or_try_init(|| py.import(name: "opentelemetry.trace").map(op: Into::into));
    let module: &Bound<'_, PyModule> = match result {…
    };

    // opentelemetry always returns a context, but maybe with an invalid span ID = 0
    let python_span_context: Bound<'_, PyAny> = module &Bound<'_, PyModule>
        .call_method0(name: intern!(py, "get_current_span"))? Bound<'_, PyAny>
        .call_method0(name: intern!(py, "get_span_context"))?;

    let trace_id: u128 = python_span_context Bound<'_, PyAny>
        .getattr(attr_name: intern!(py, "trace_id"))? Bound<'_, PyAny>
        .extract::<u128>()?;

    if trace_id == 0 {
        return Ok(None);
    }
```

6 – Observability

```rust
pub fn get_python_opentelemetry_span_context(
    py: Python,
) -> PyResult<Option<opentelemetry::trace::SpanContext>> {
    static OPENTELEMETRY_TRACE_IMPORT: OnceCell<Py<PyModule>> = OnceCell::new();

    let result: Result<&Py<PyModule>, PyErr> = OPENTELEMETRY_TRACE_IMPORT OnceCell<Py<P…
        .get_or_try_init(|| py.import(name: "opentelemetry.trace").map(op: Into::into));
    let module: &Bound<'_, PyModule> = match result {…
    };

    // opentelemetry always returns a context, but maybe with an invalid span ID = 0
    let python_span_context: Bound<'_, PyAny> = module &Bound<'_, PyModule>
        .call_method0(name: intern!(py, "get_current_span"))? Bound<'_, PyAny>
        .call_method0(name: intern!(py, "get_span_context"))?;

    let trace_id: u128 = python_span_context Bound<'_, PyAny>
        .getattr(attr_name: intern!(py, "trace_id"))? Bound<'_, PyAny>
        .extract::<u128>()?;

    if trace_id == 0 {
        return Ok(None);
    }
```

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

```rust
pub fn set_python_span_as_parent(py: Python) {
    let span: &Span = &tracing::Span::current();
    if span.is_disabled() {
        return;
    }

    let span_context: Result<Option<SpanContext>, …> = get_python_opentelemetry_span_context(py);

    if let Ok(Some(span_context: SpanContext)) = span_context {
        let context: Context = opentelemetry::Context::new().with_remote_span_context(span_context);
        span.set_parent(cx: context);
    }
}
```

7 – From PoC to Production

Build Rust using the release profile

Version compatibility: PyO3 – rustc – python


FEATURE FLAGS
FEATURE FLAGS EVERYWHERE

7 – From PoC to Production

7 – From PoC to Production


THIS IS FINE

7 – From PoC to Production

8 – The End

2023

170K tasks/day

380K all time high

2025

340K tasks/day

978K all time high

8 – The End

Success can breed failure

~~Silver bullet~~

Focus on understanding and fixing the underlying root cause

Use existing toolset to its fullest potential

Anything that can go wrong, will go wrong

Use Rust where parallelism matters

Test as close to production as possible

Build Rust using the release profile

Release the GIL

2023

170K tasks/day

380K all time high

2025

340K tasks/day

978K all time high

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

Version compatibility: PyO3 – rustc – python

Scale matters – rust integration comes at a cost

Keep legacy implementation as fallback

Future proof your APIs by focussing on how they might break

Integration tests, integration tests and more integration tests