

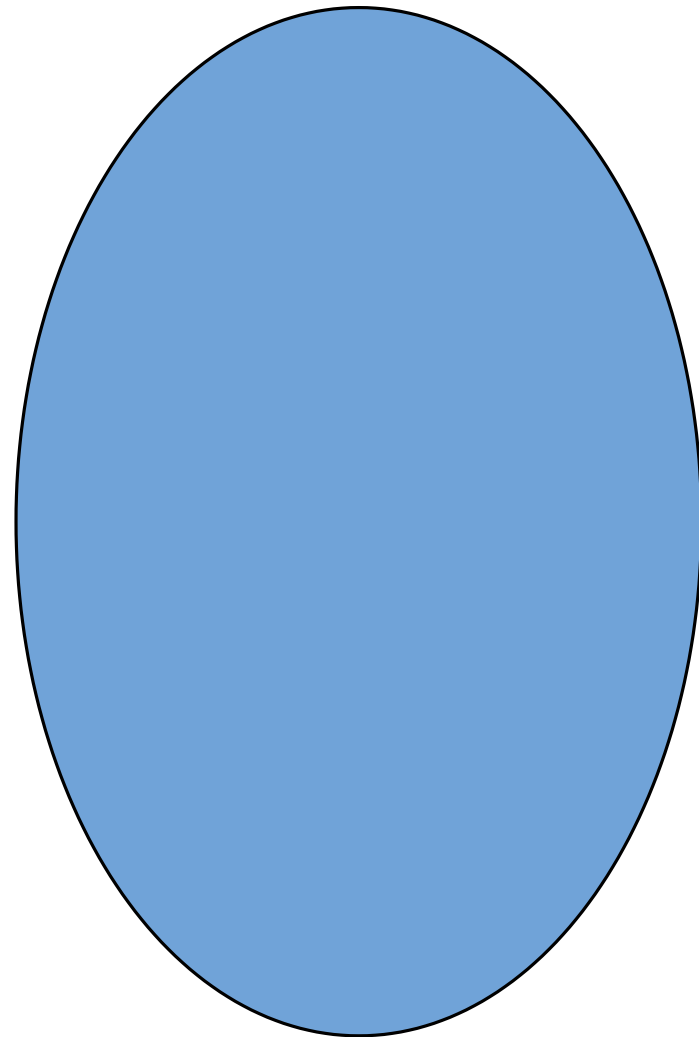


ANALYSIS OF GLOBAL TEMPERATURE

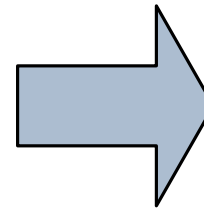
PART 3

Sum Temperature

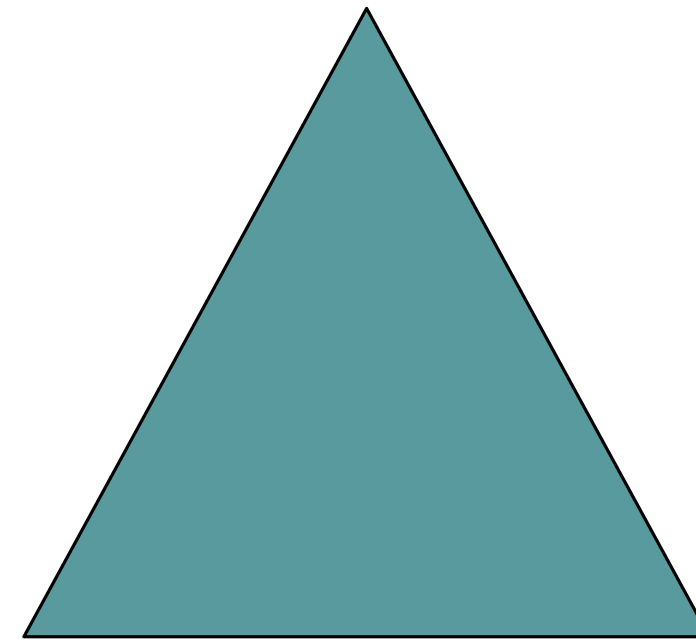
ParList[Sample]



monoFoldLeft

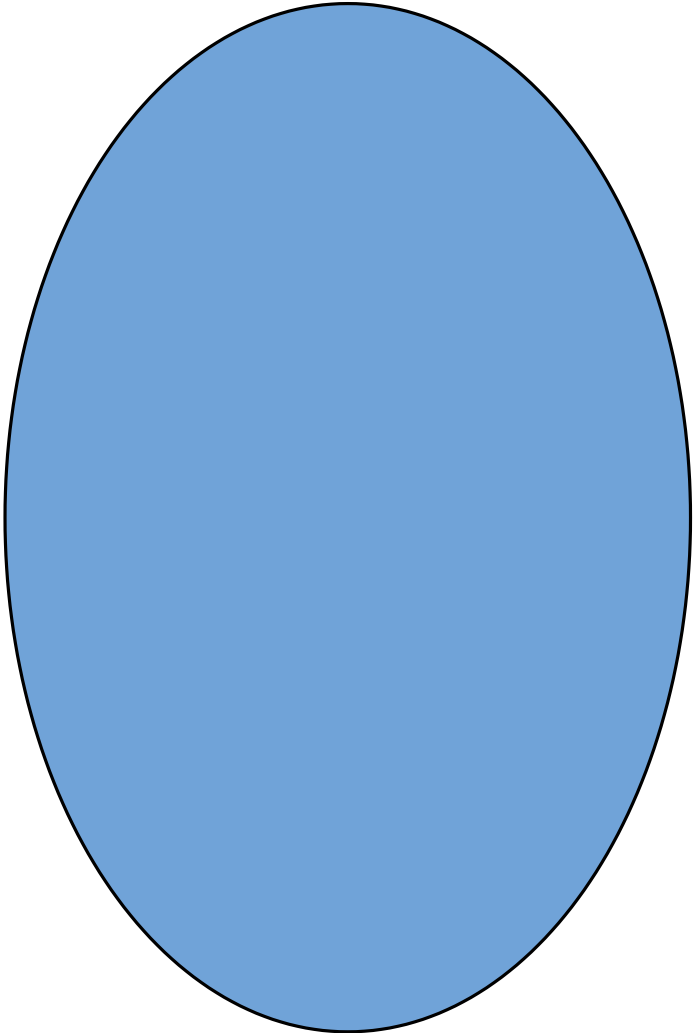


Double

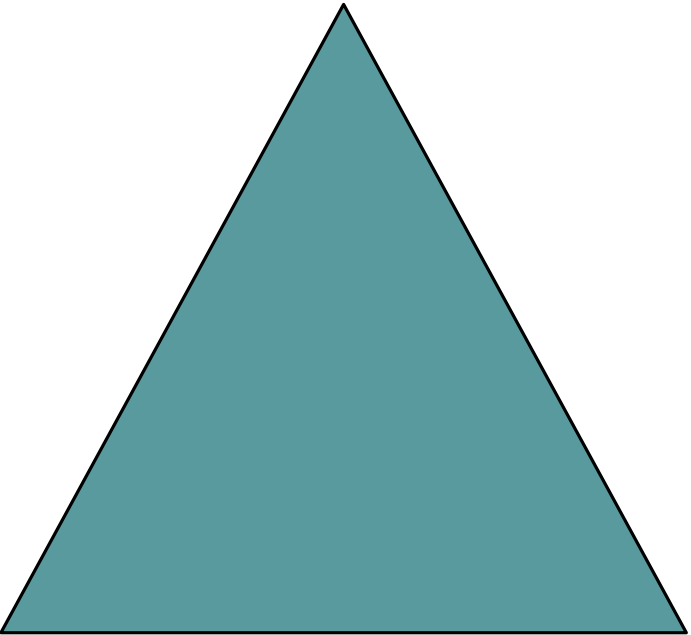


Sum Temperature

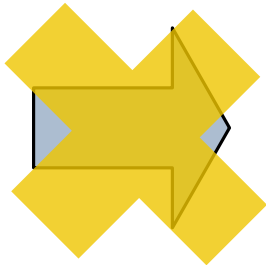
ParList[Sample]



Double

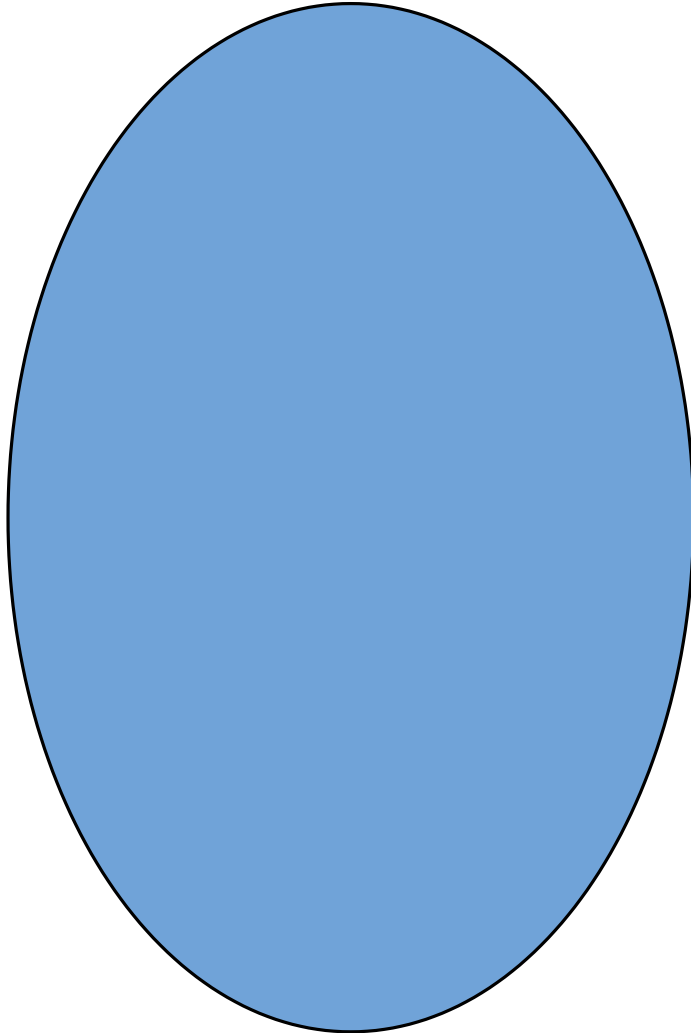


monoFoldLeft

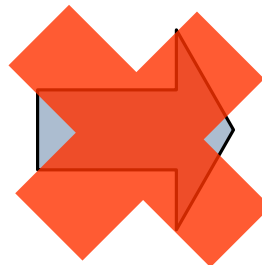


Sum Temperature

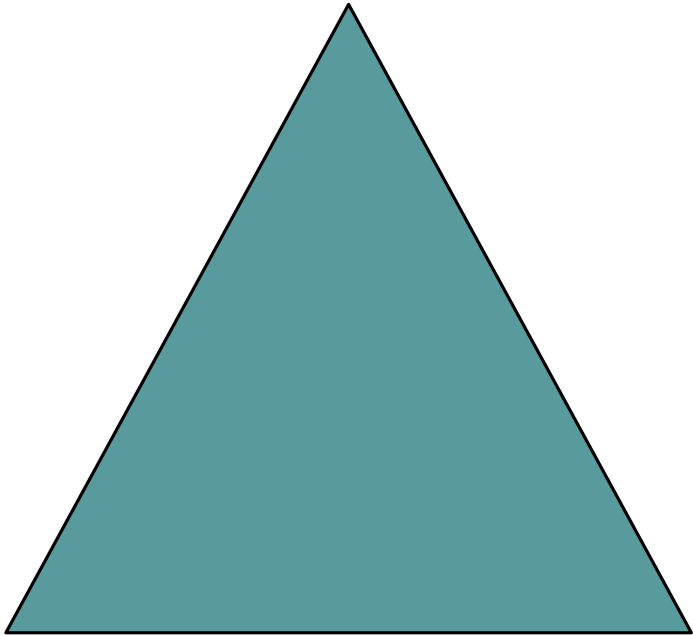
ParList[Sample]



monoFoldLeft

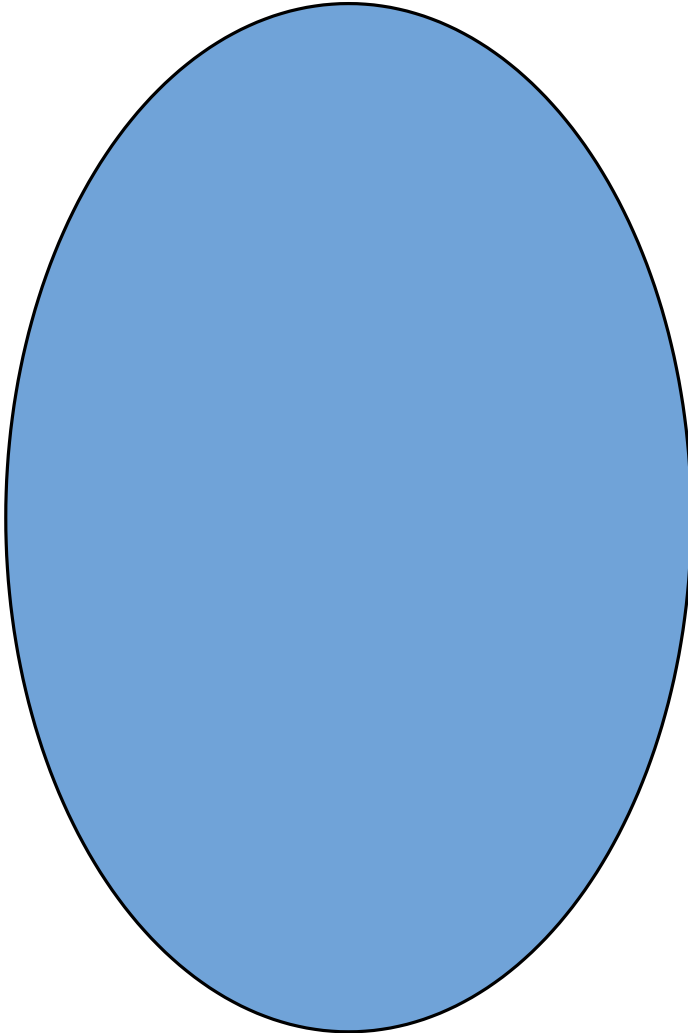


Double

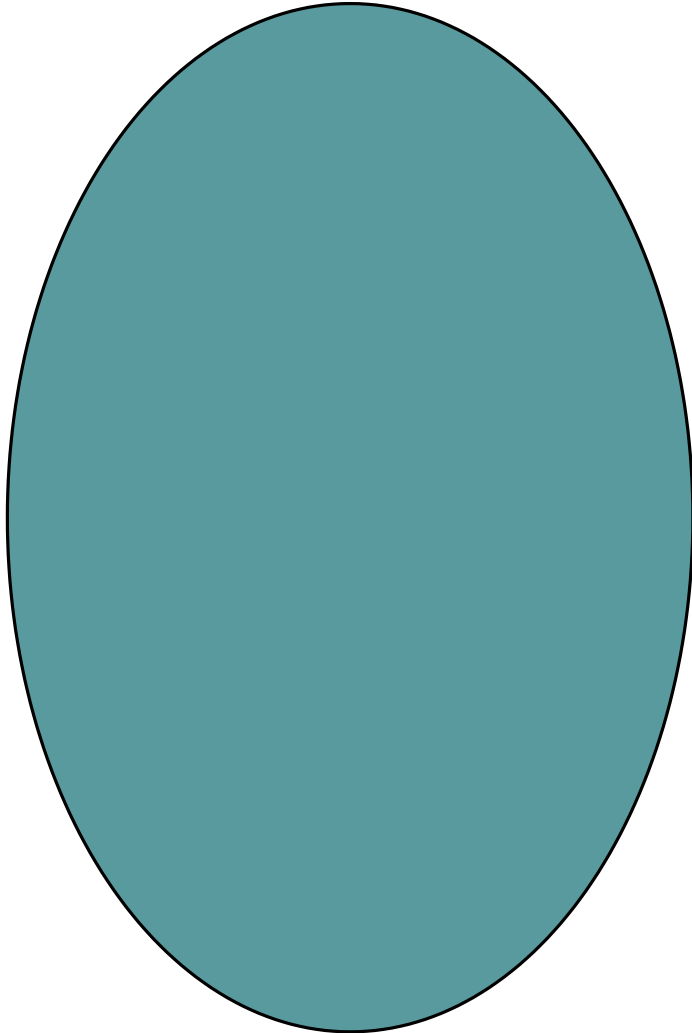


Sum Temperature

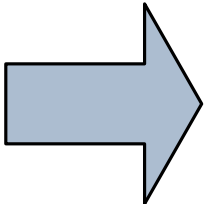
ParList[Sample]



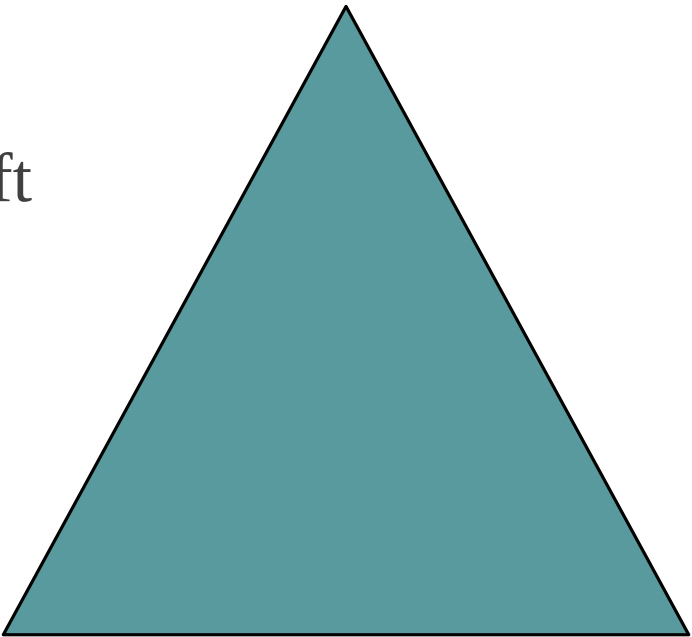
ParList[Double]



monoFoldLeft

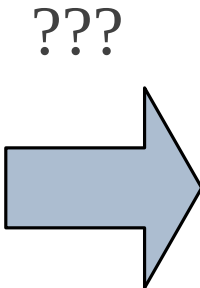
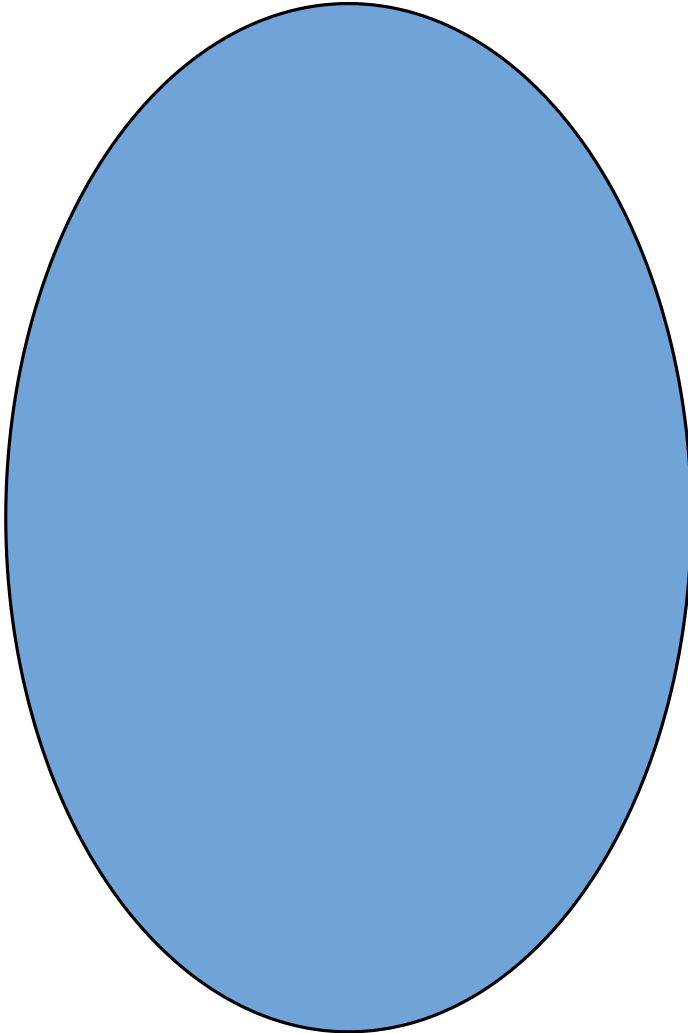


Double

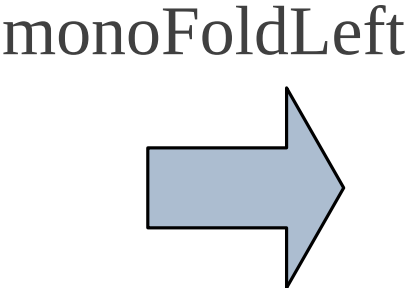
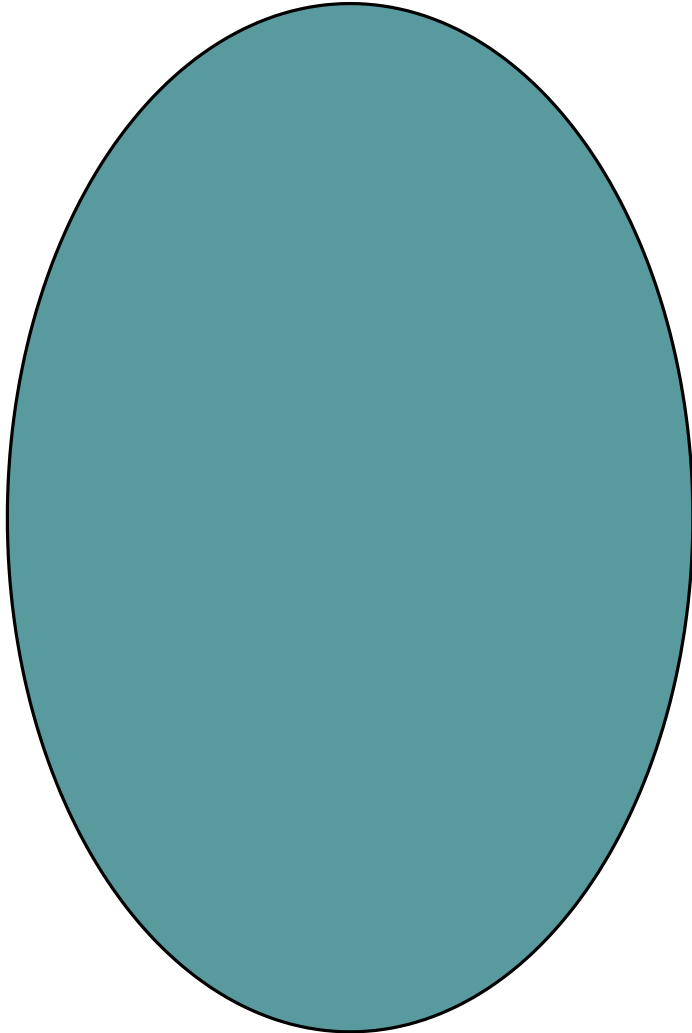


Sum Temperature

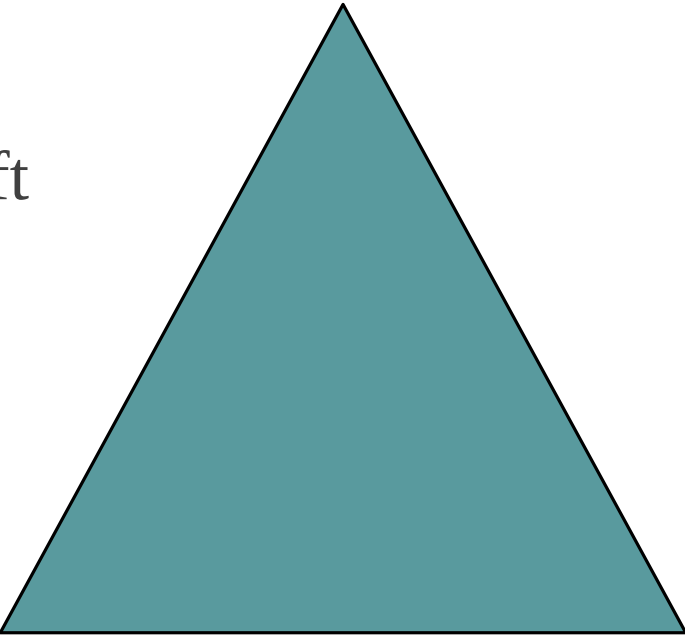
ParList[Sample]



ParList[Double]

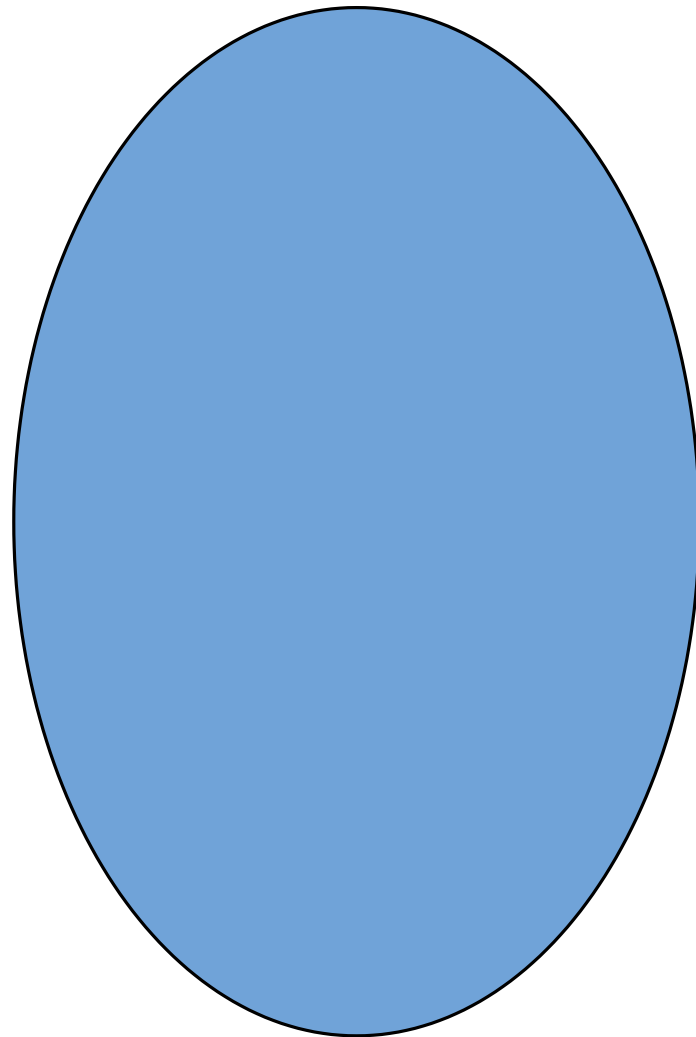


Double

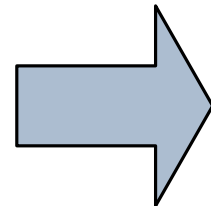


Sum Temperature

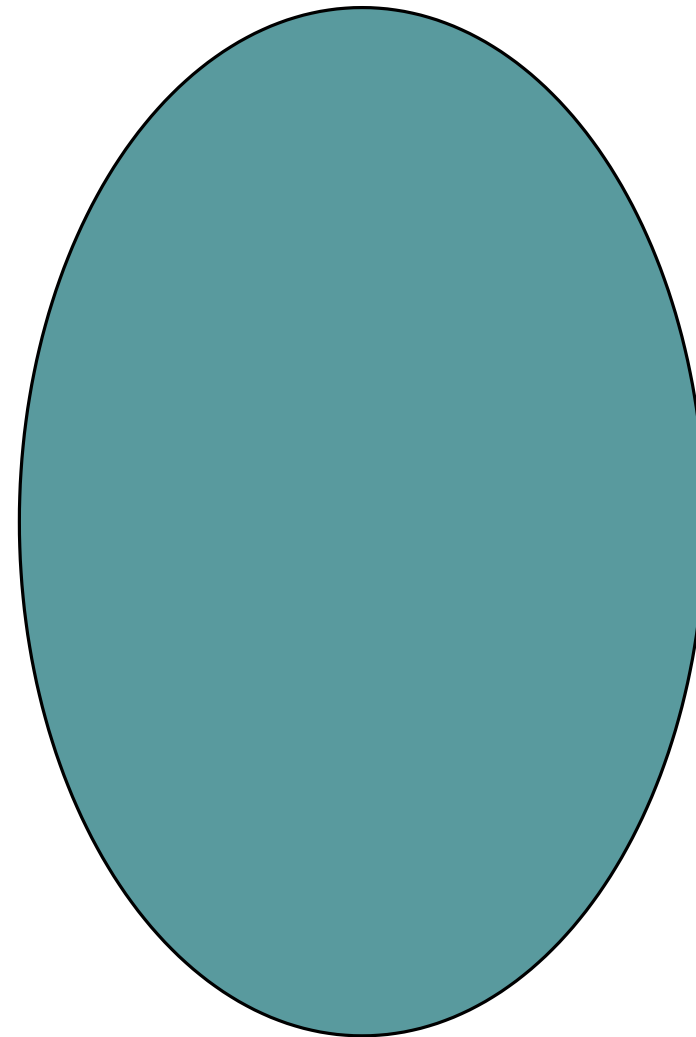
ParList[Sample]



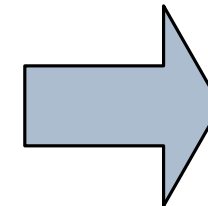
map



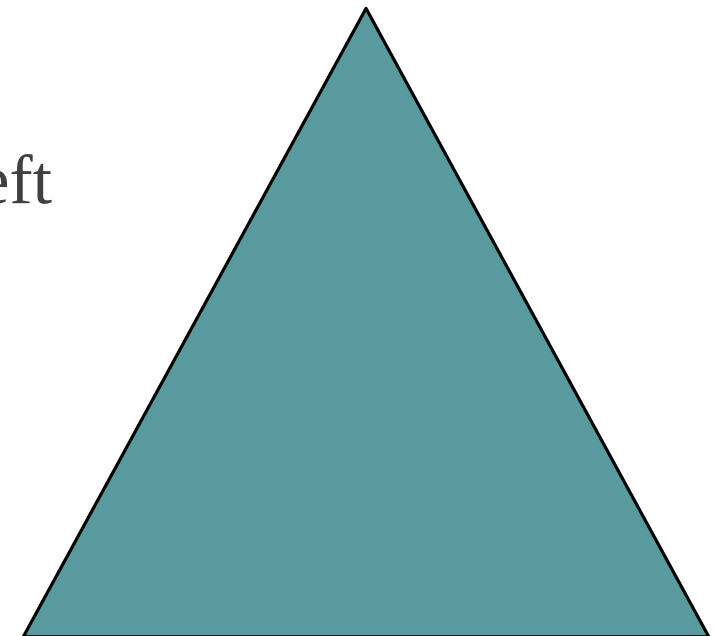
ParList[Double]



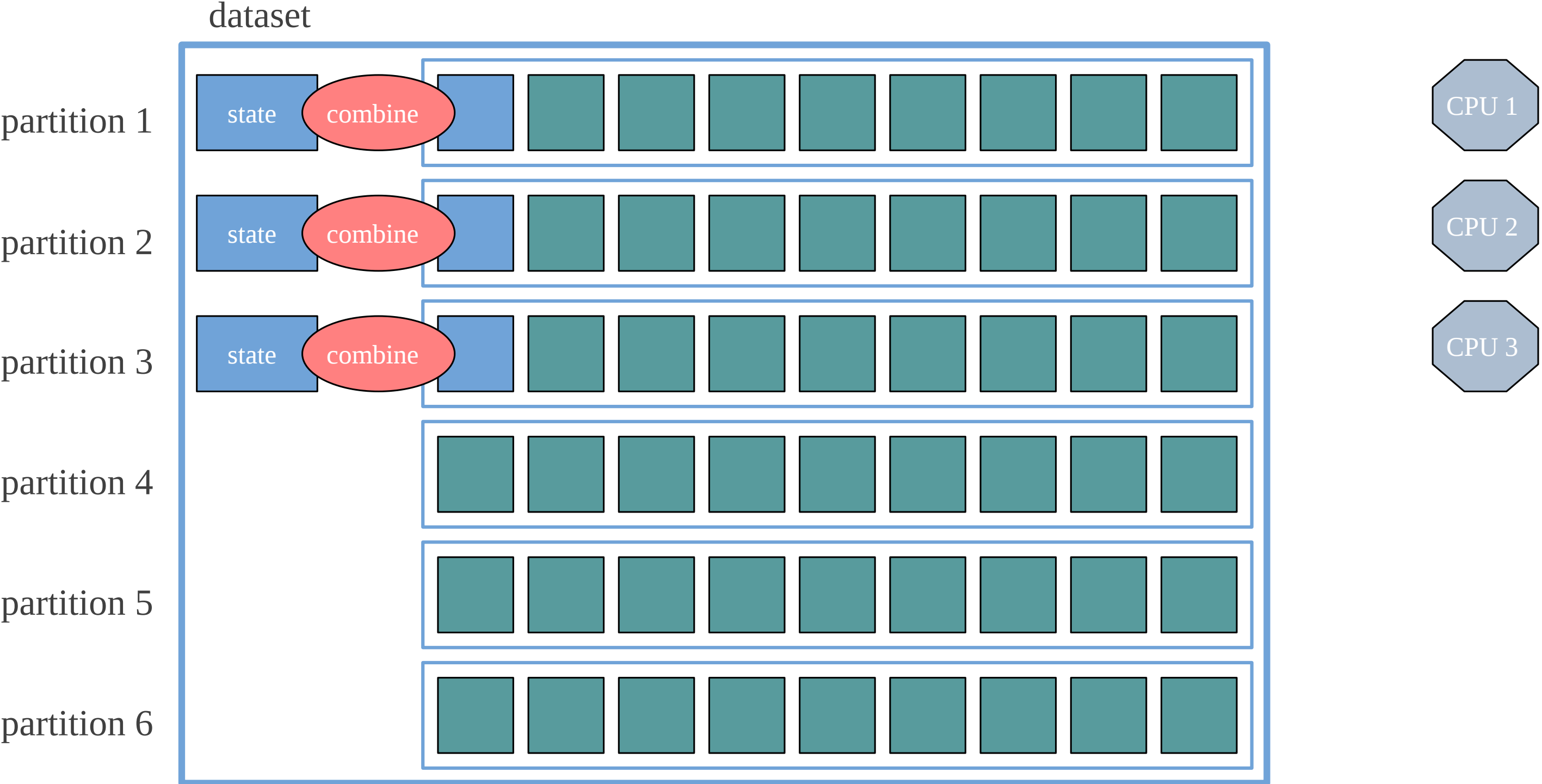
monoFoldLeft



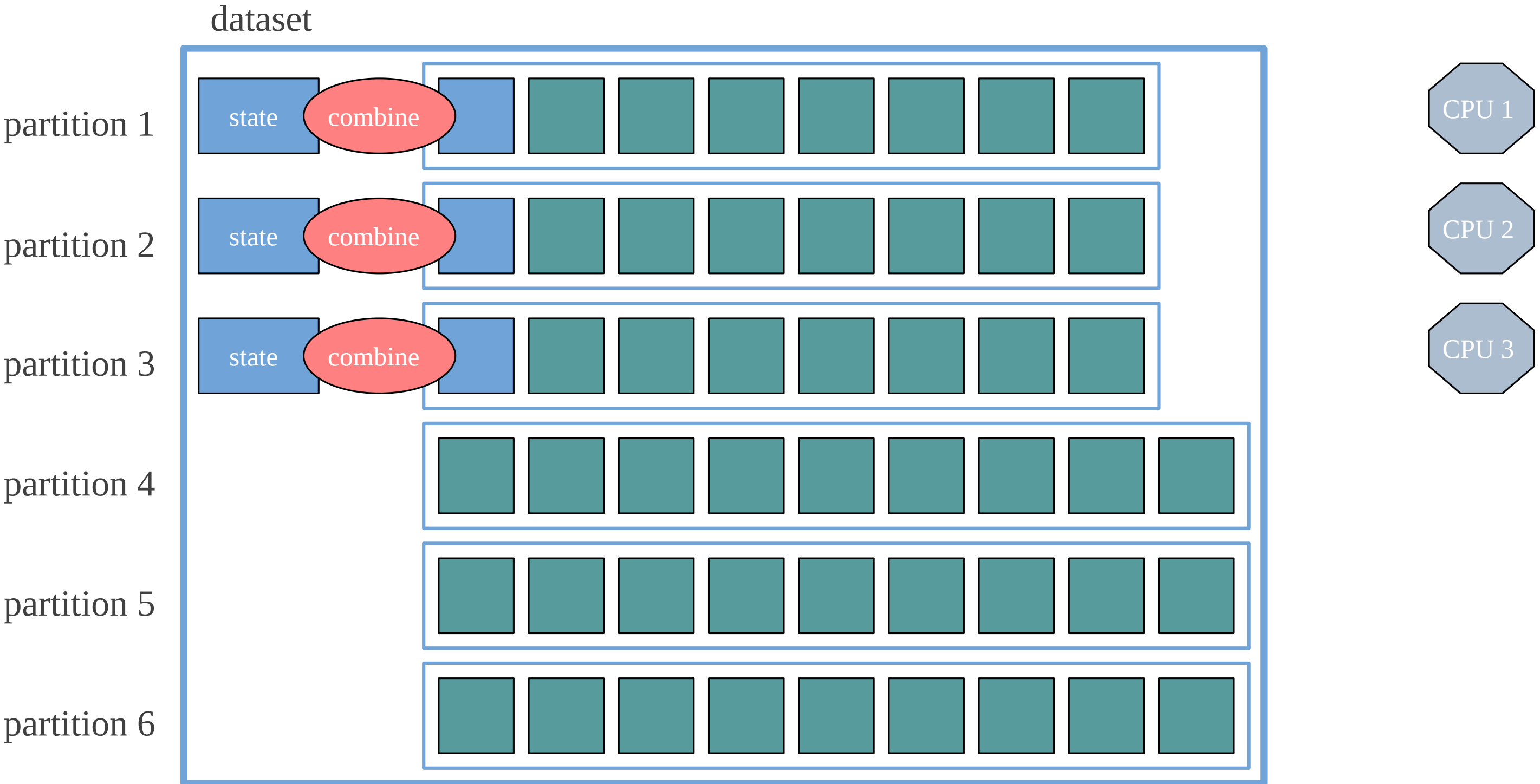
Double



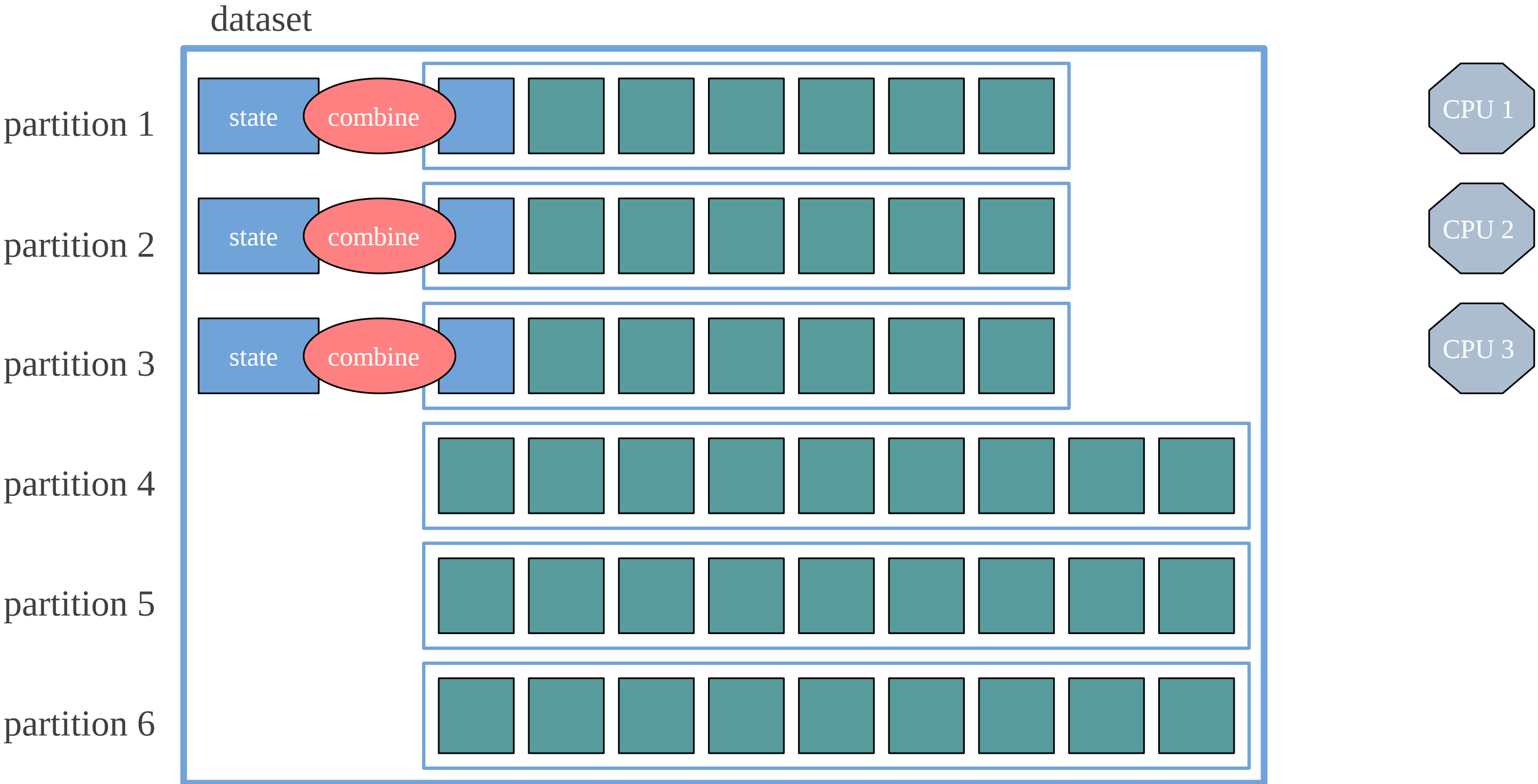
foldMap in parallel



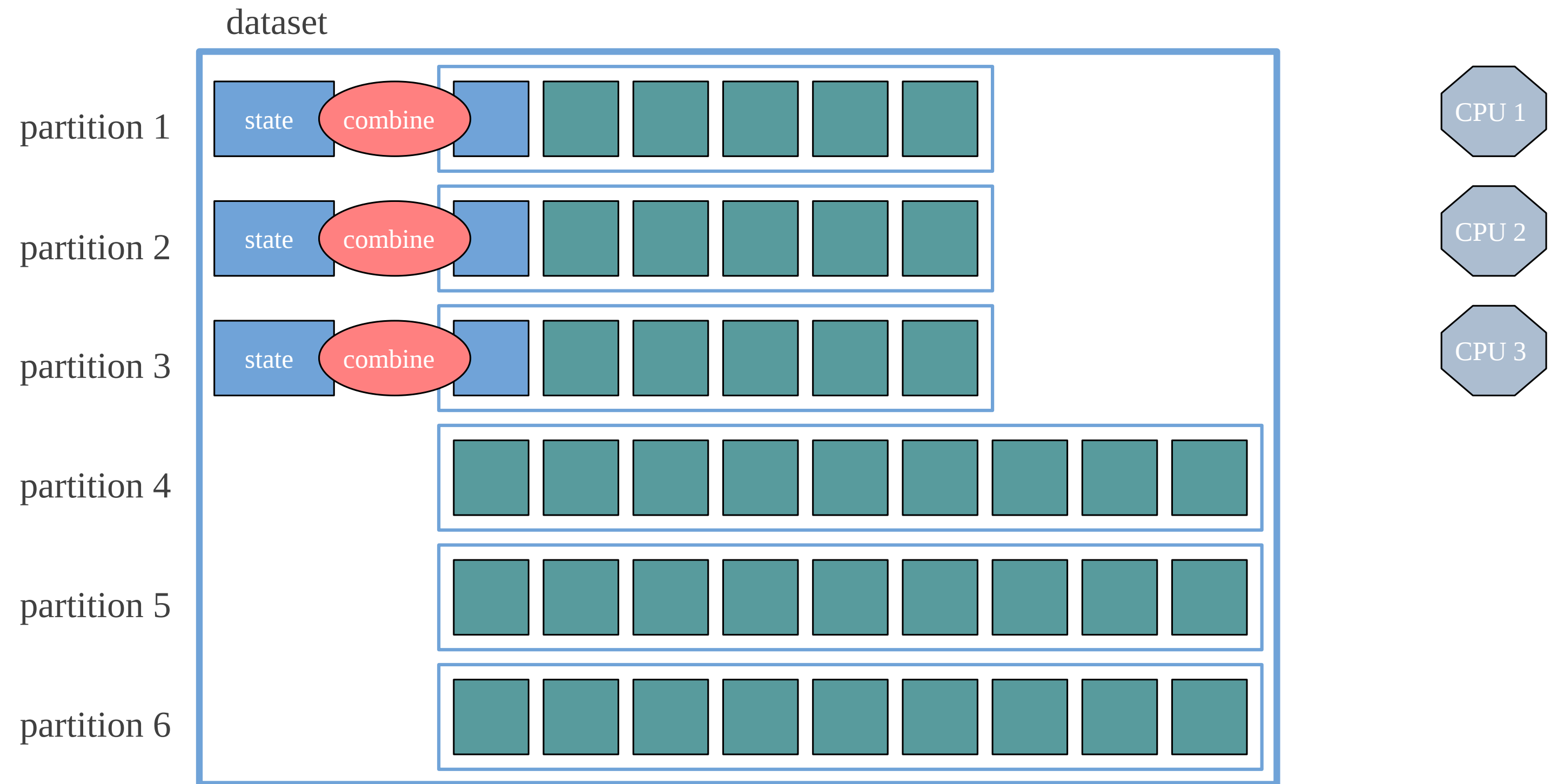
foldMap in parallel



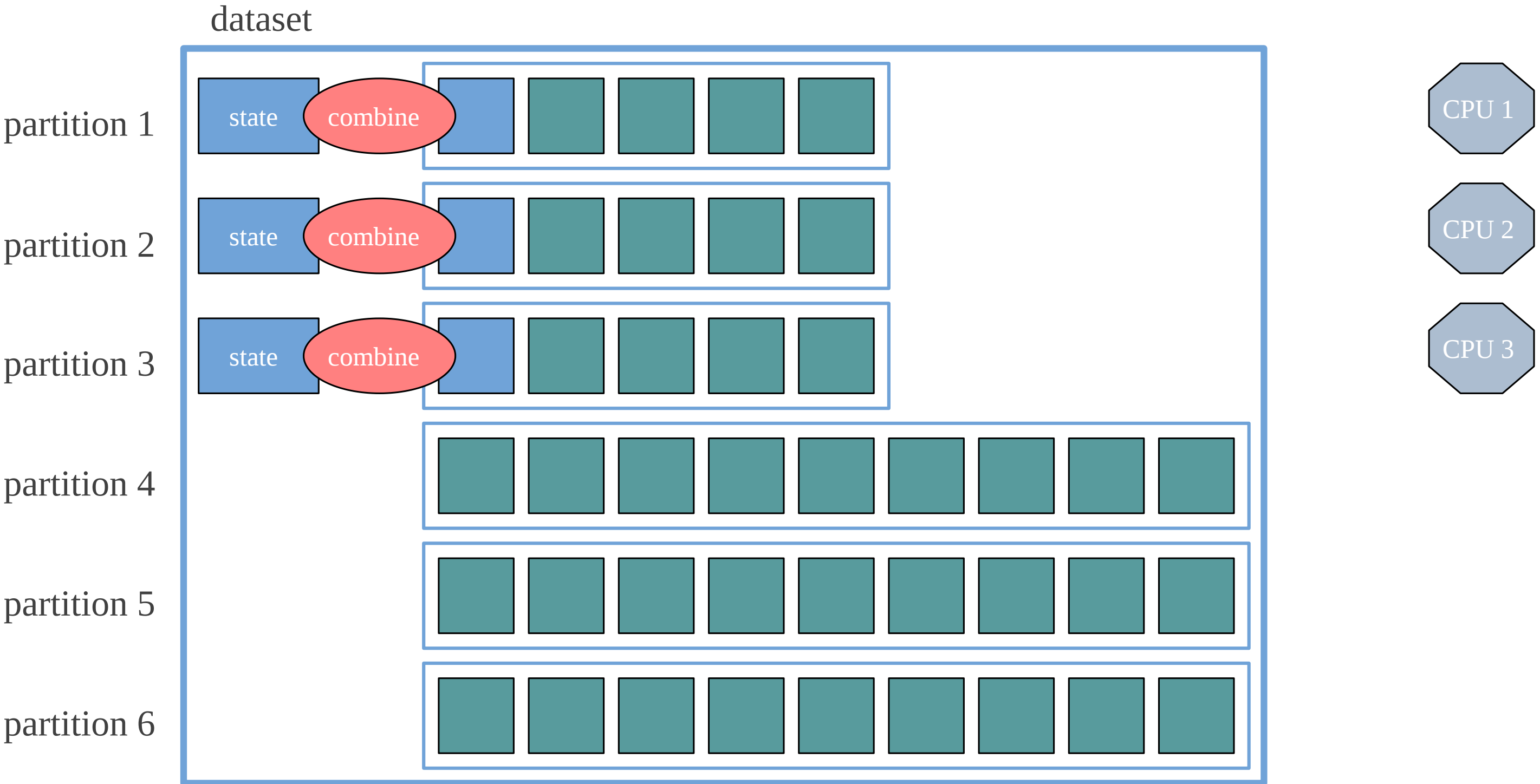
foldMap in parallel



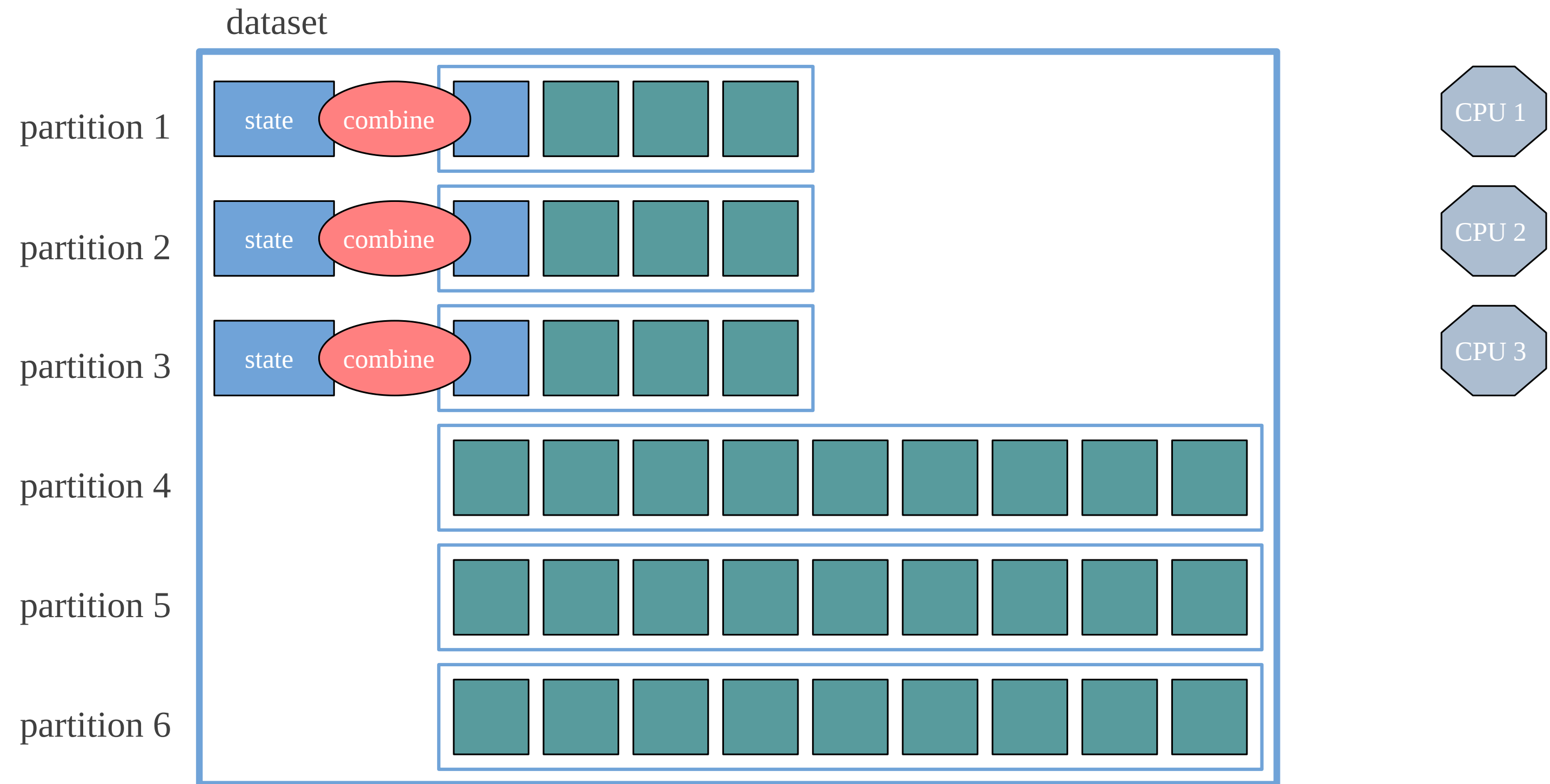
foldMap in parallel



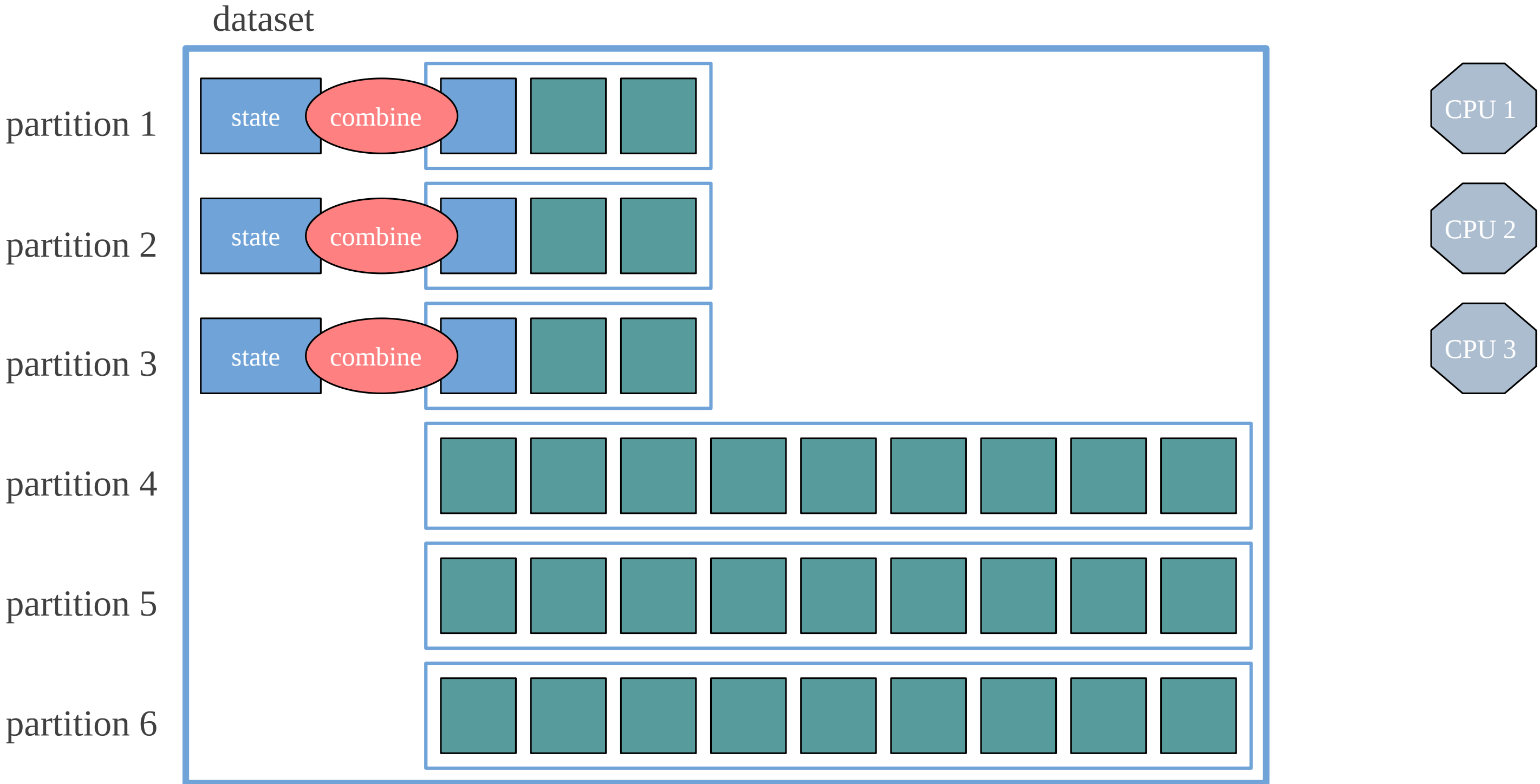
foldMap in parallel



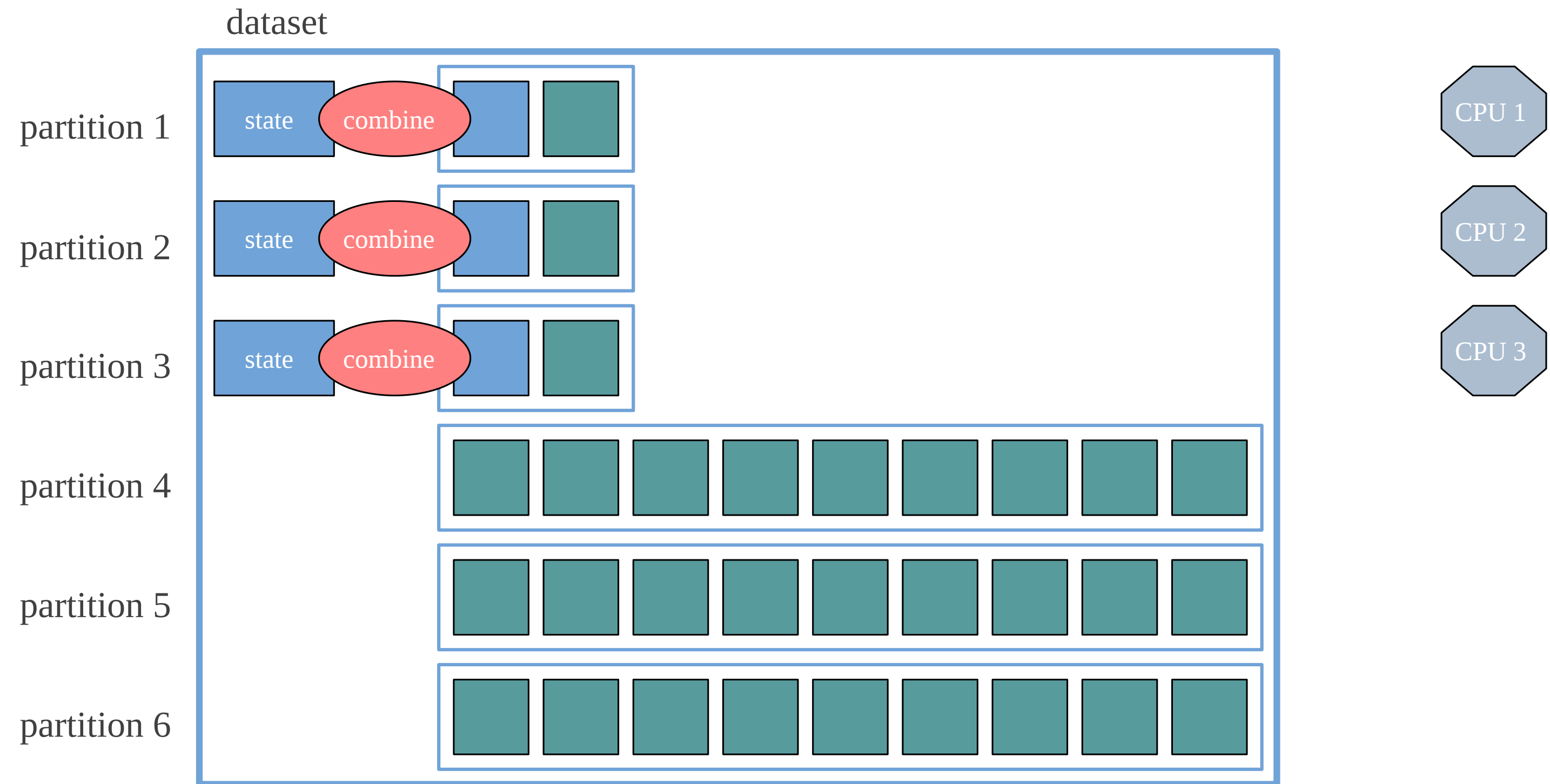
foldMap in parallel



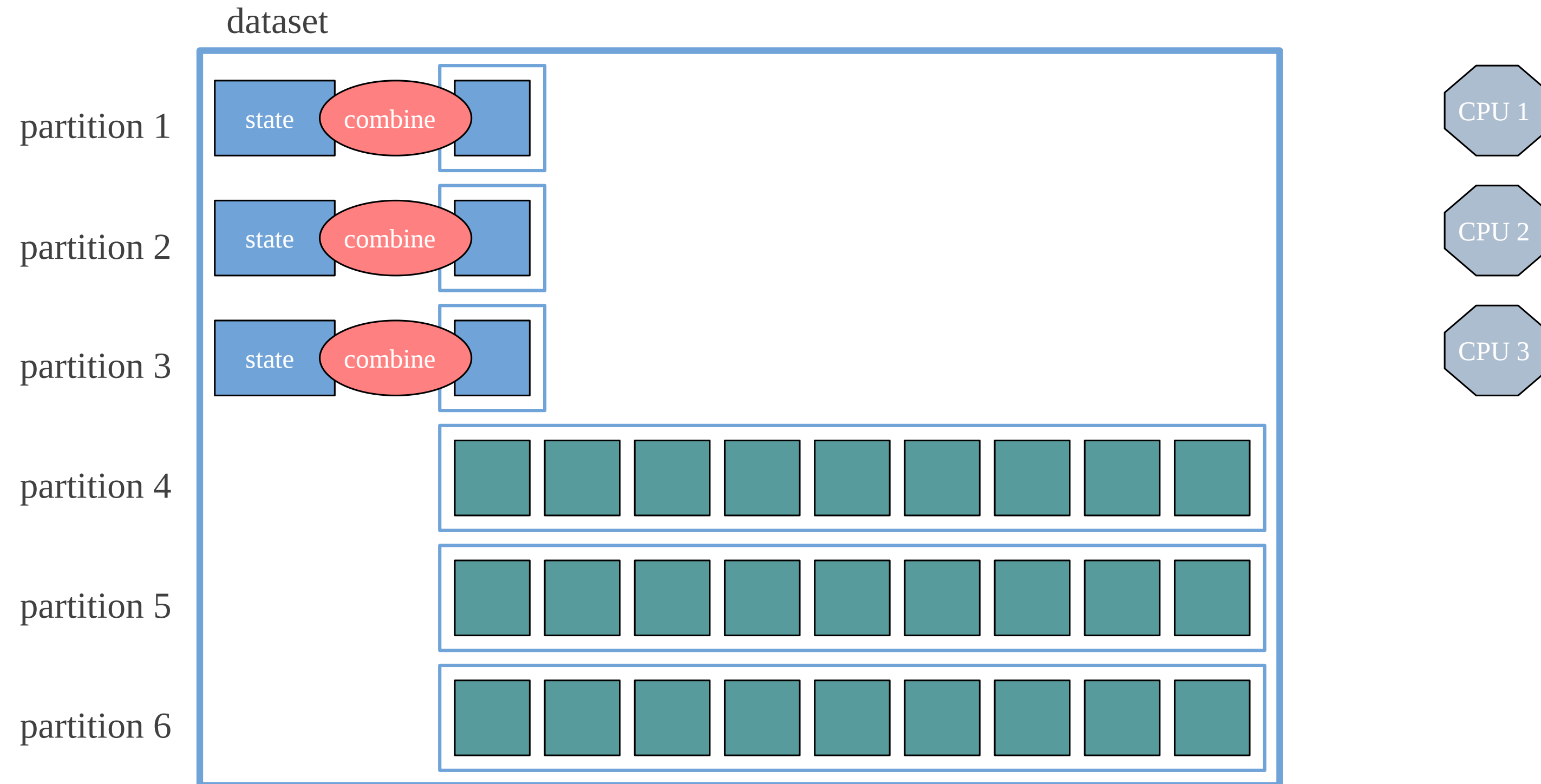
foldMap in parallel



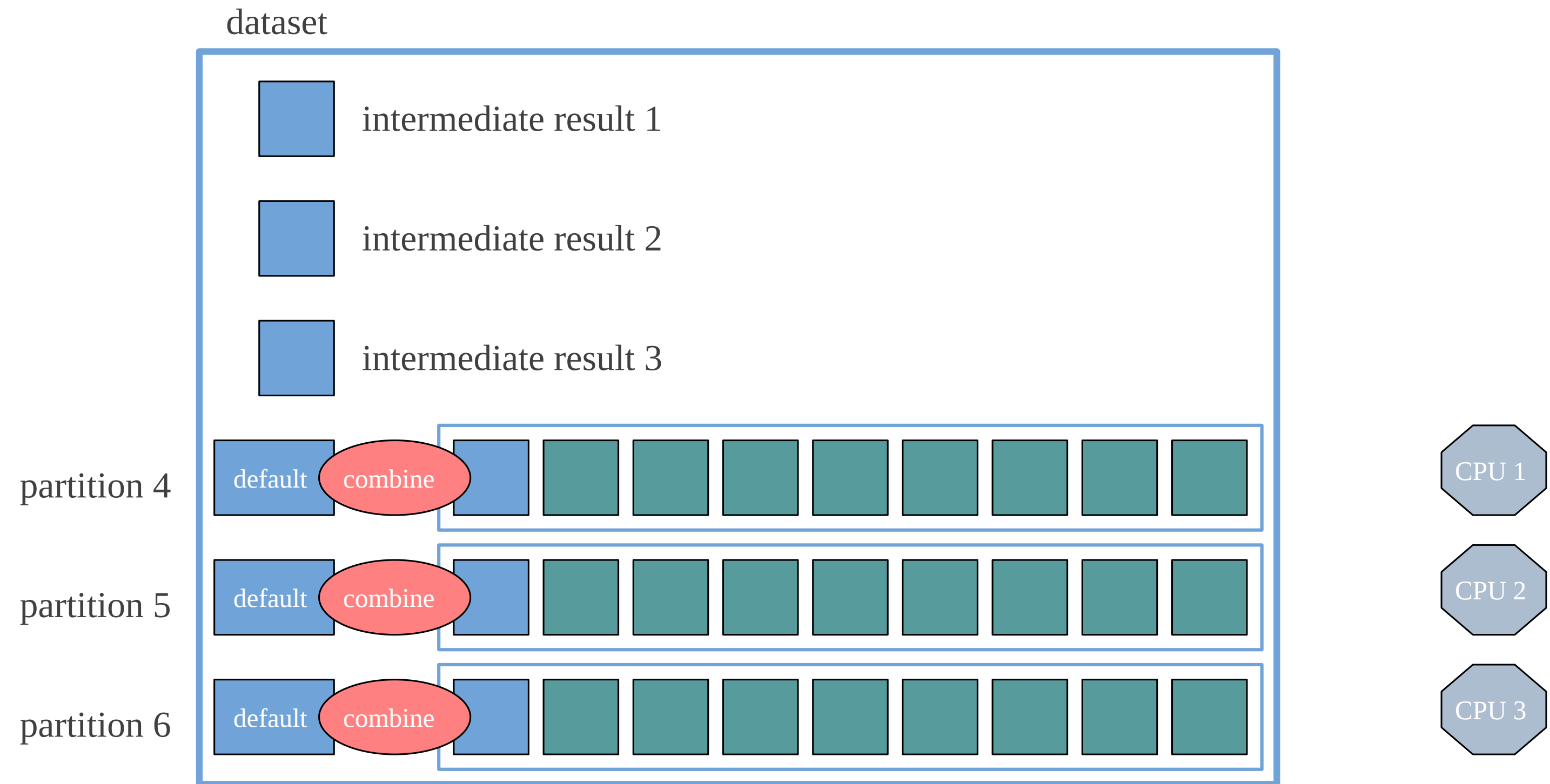
foldMap in parallel



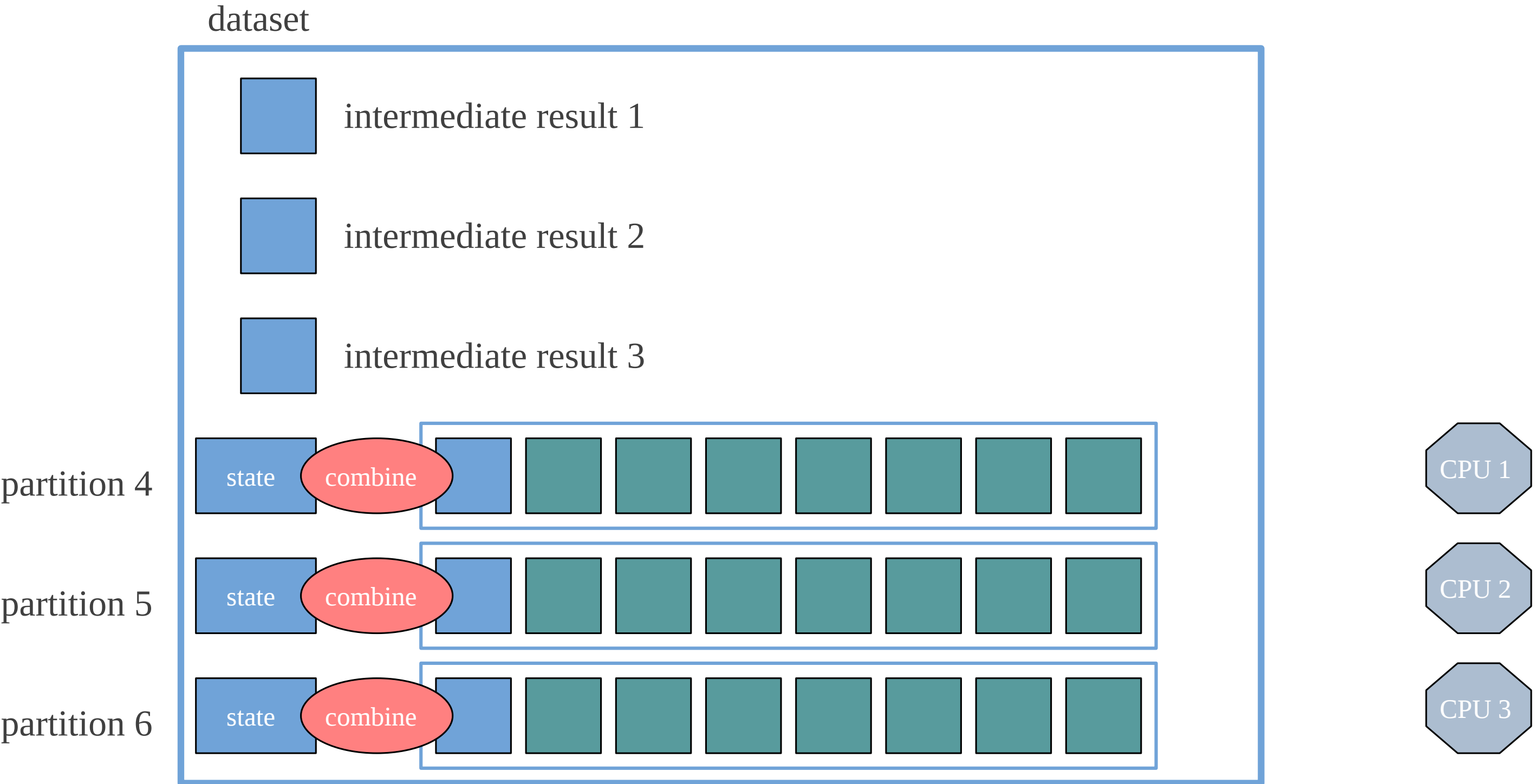
foldMap in parallel



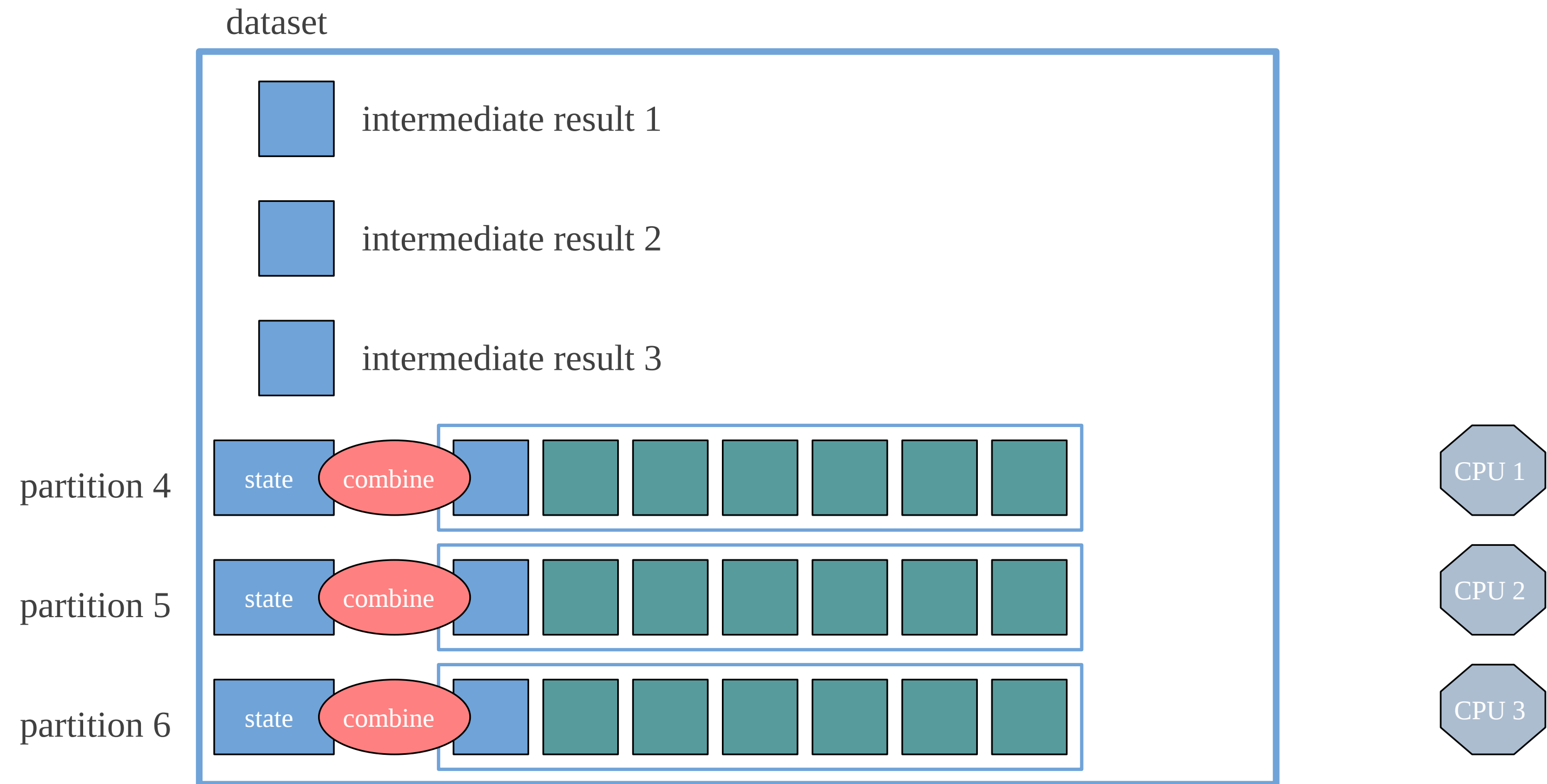
foldMap in parallel



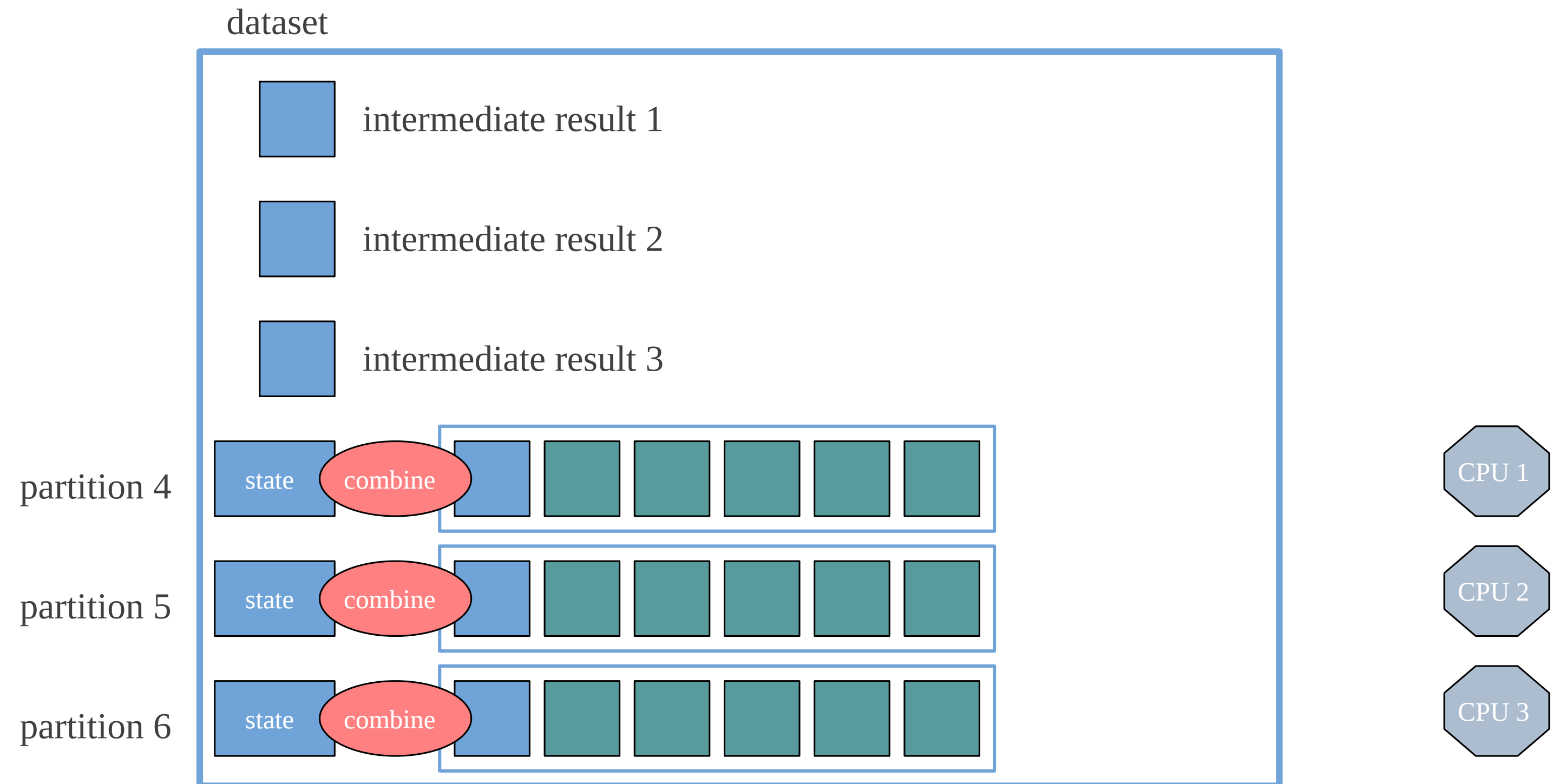
foldMap in parallel



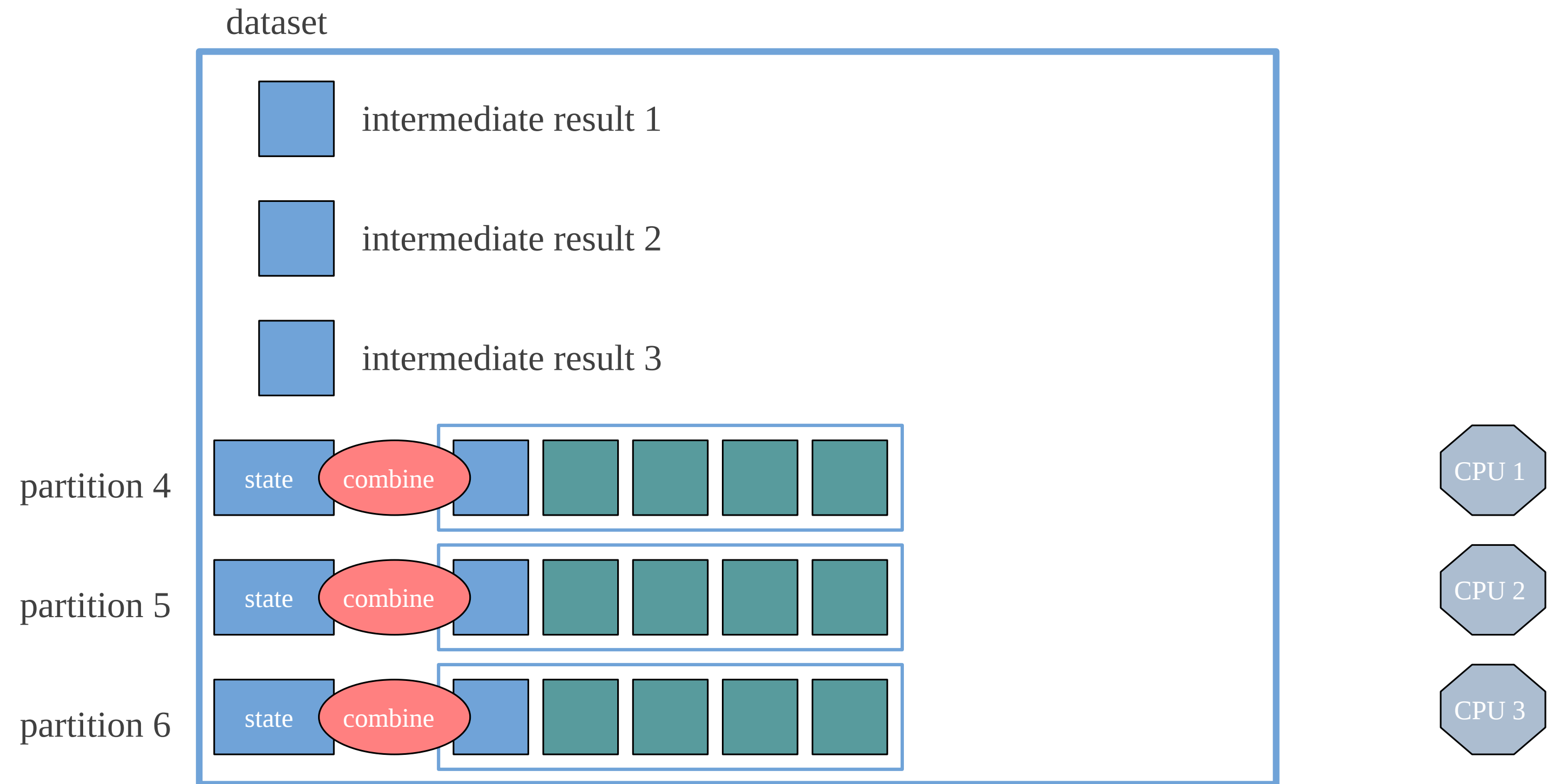
foldMap in parallel



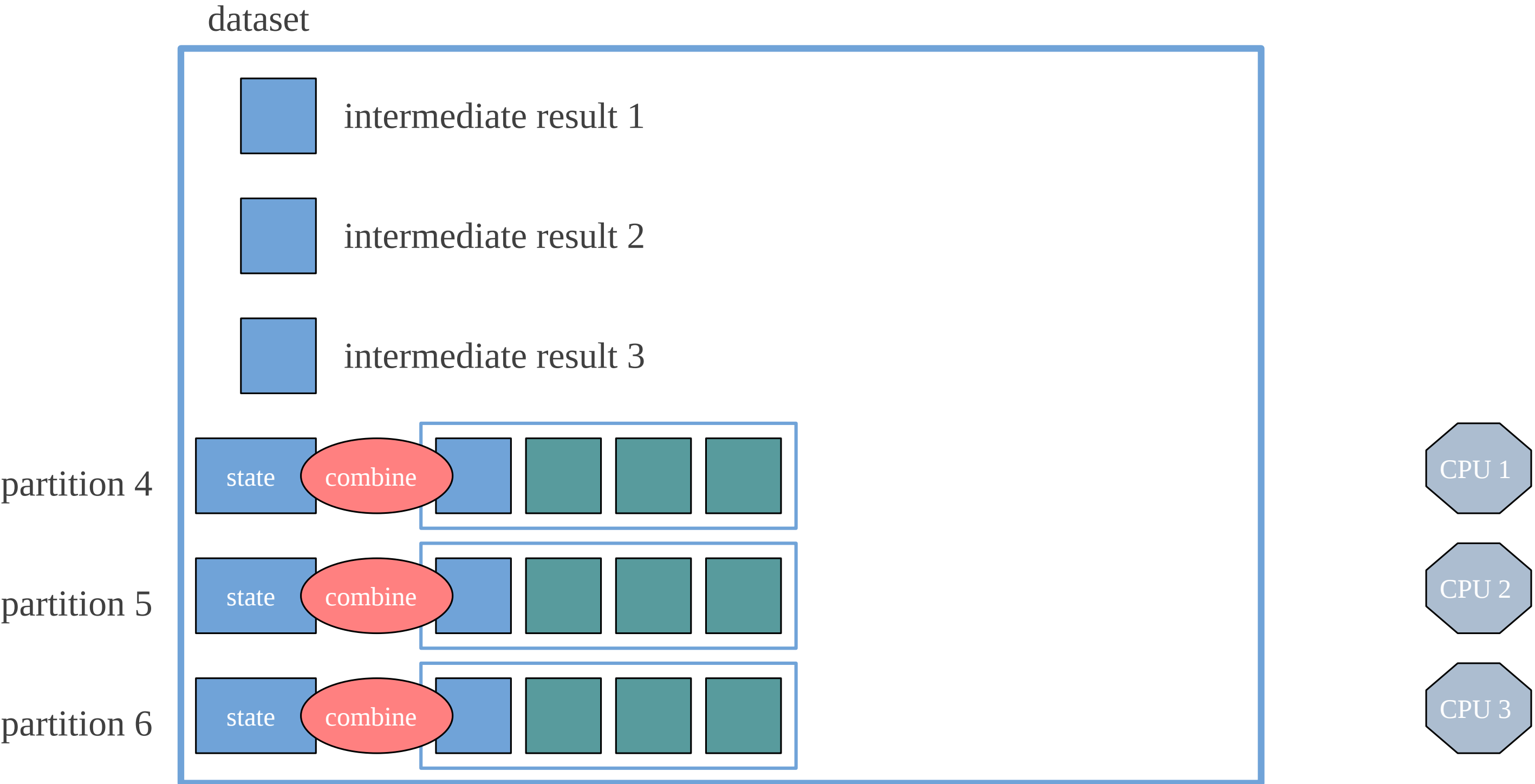
foldMap in parallel



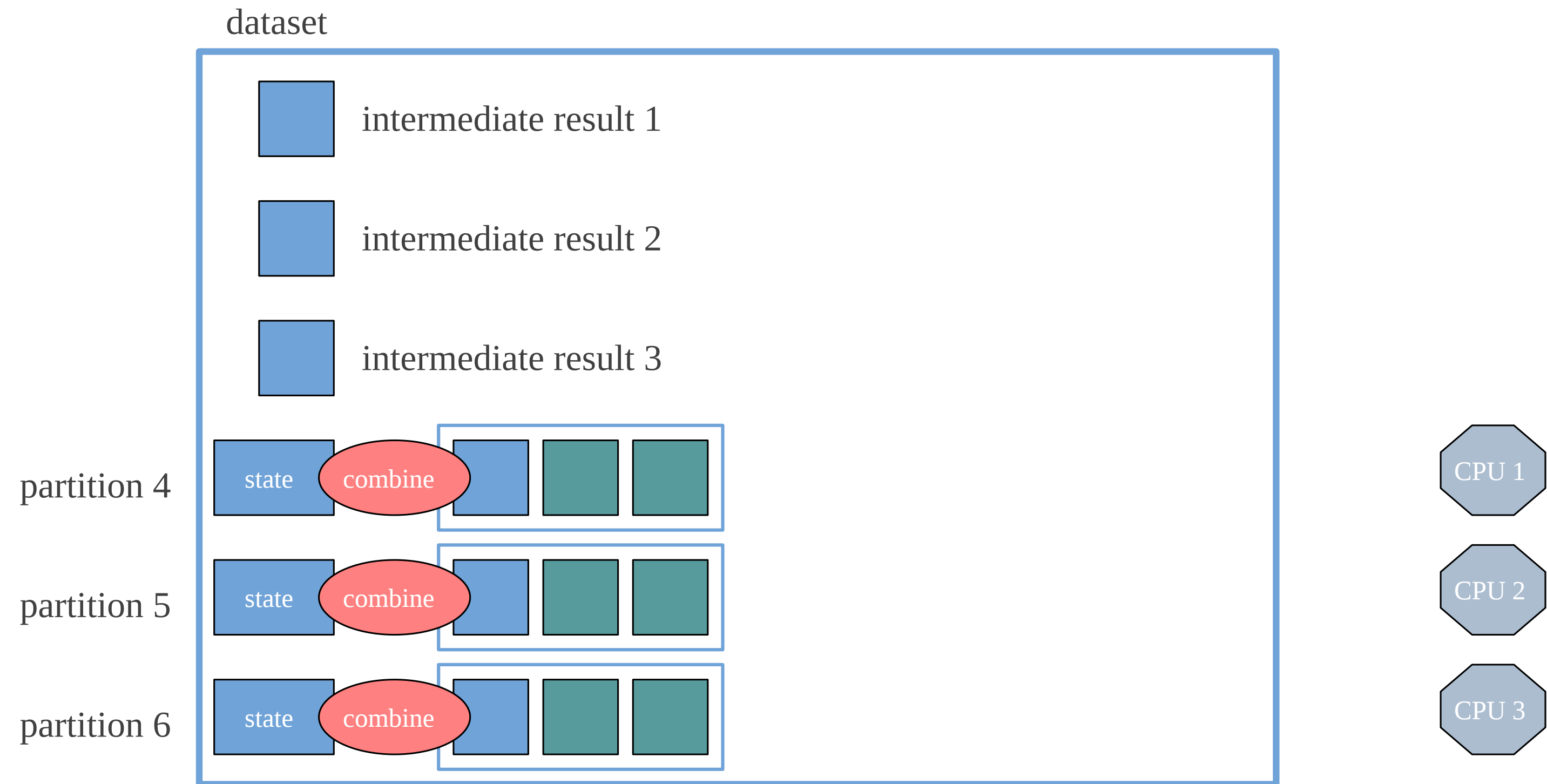
foldMap in parallel



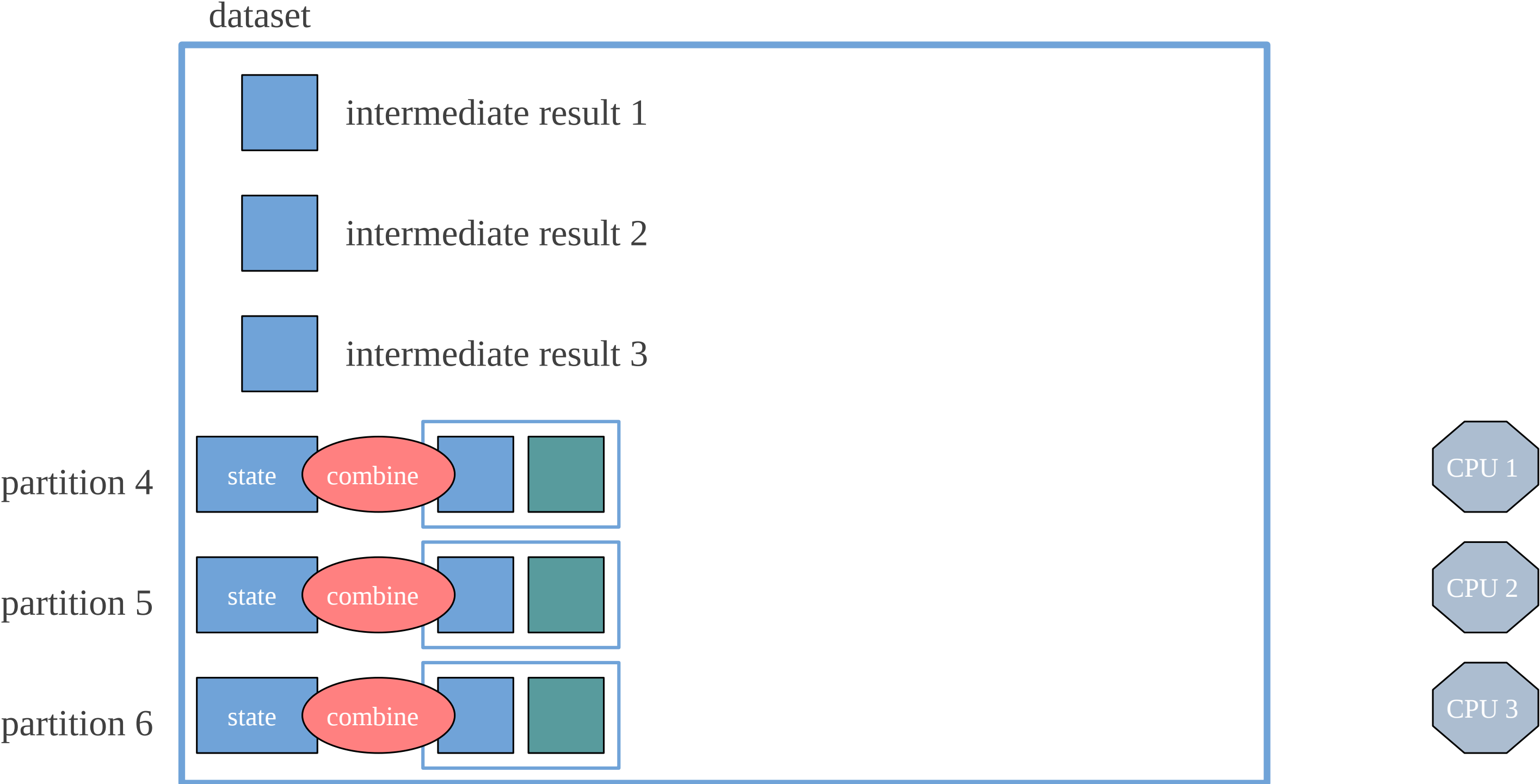
foldMap in parallel



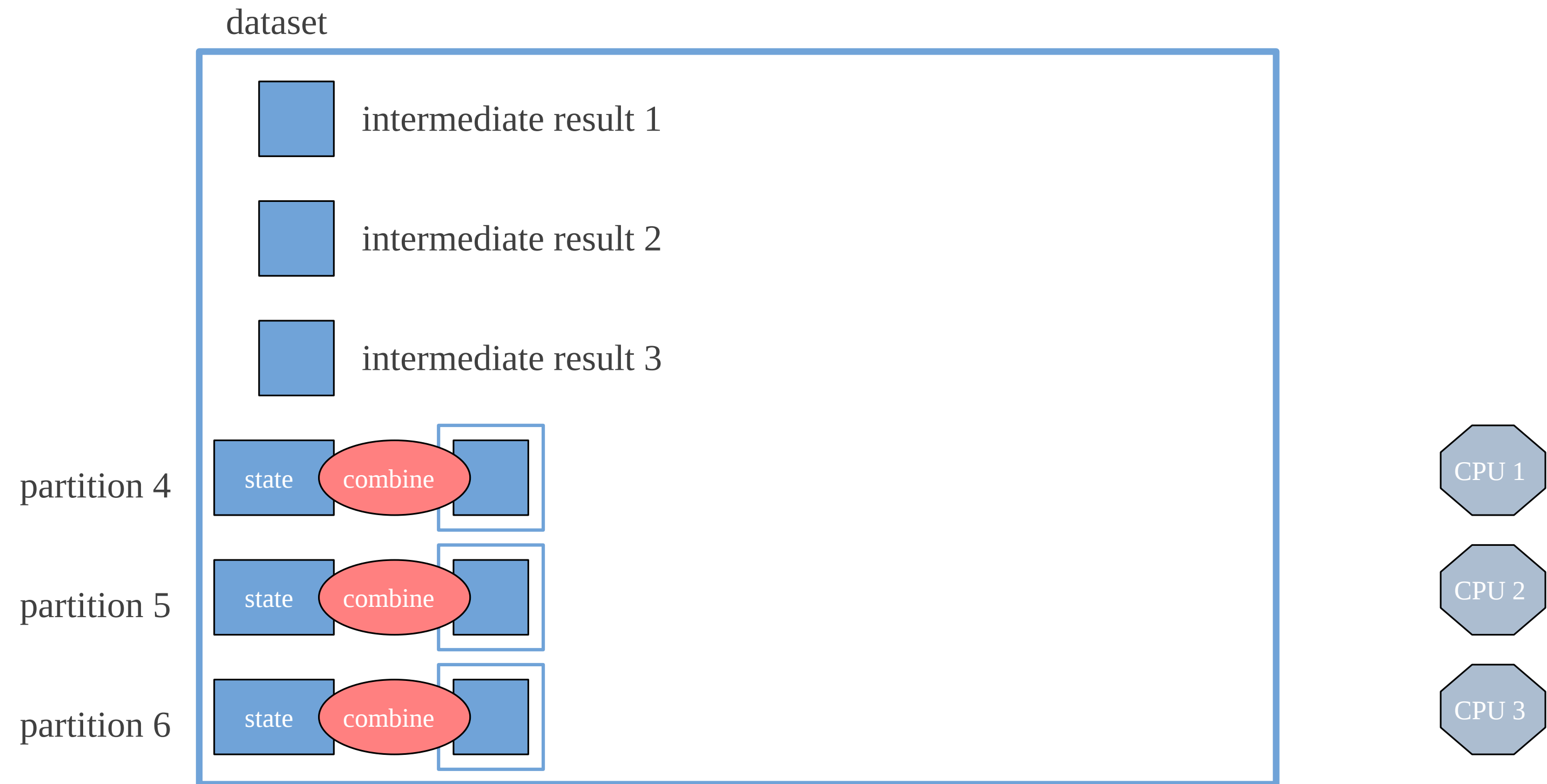
foldMap in parallel



foldMap in parallel

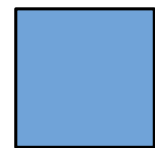


foldMap in parallel

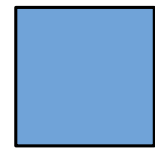


foldMap in parallel

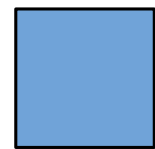
dataset



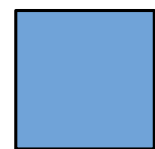
intermediate result 1



intermediate result 2



intermediate result 3



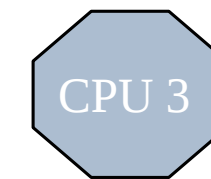
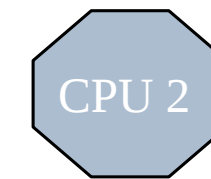
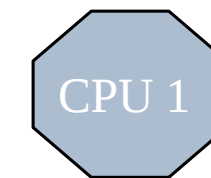
intermediate result 4



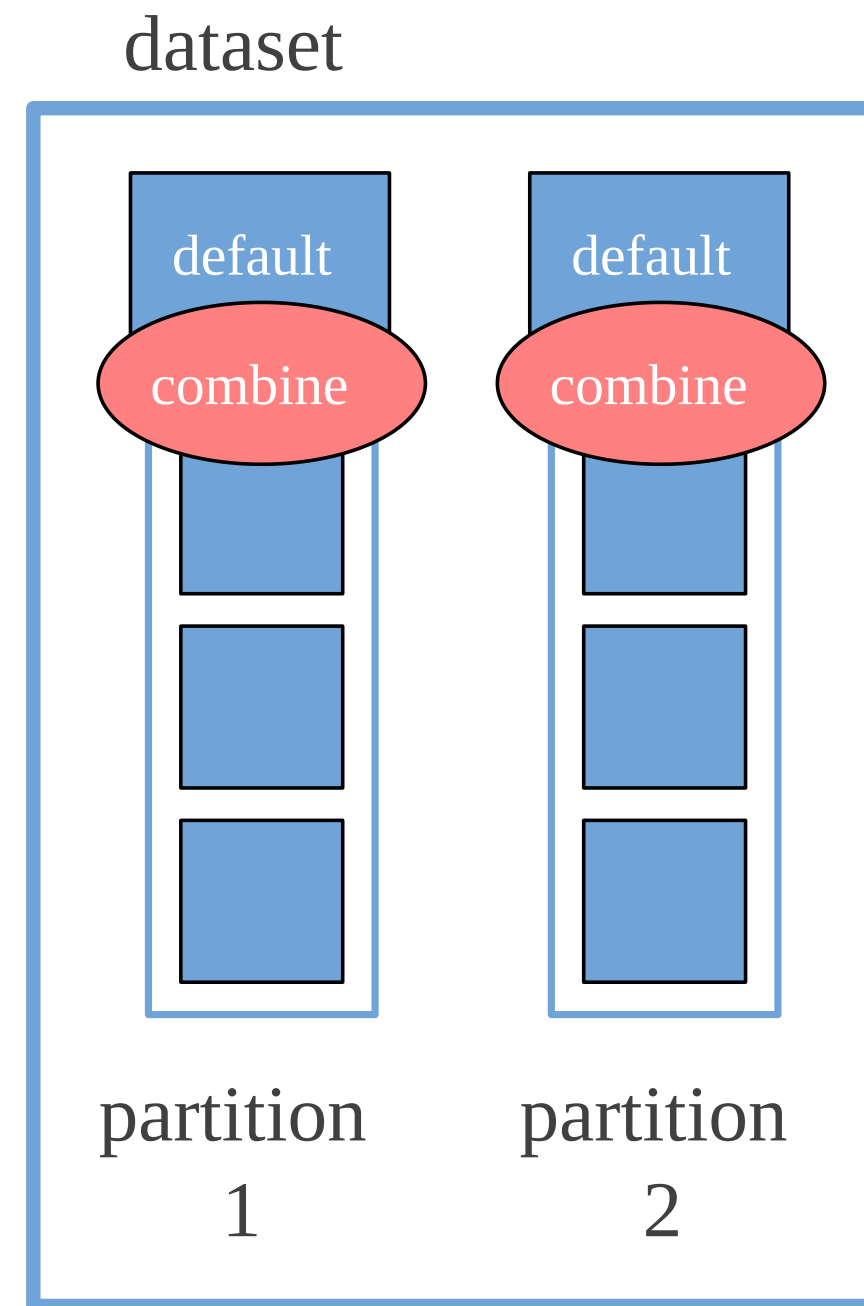
intermediate result 5



intermediate result 6



fold intermediate results in parallel

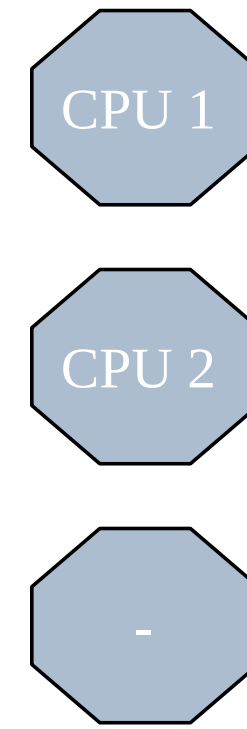
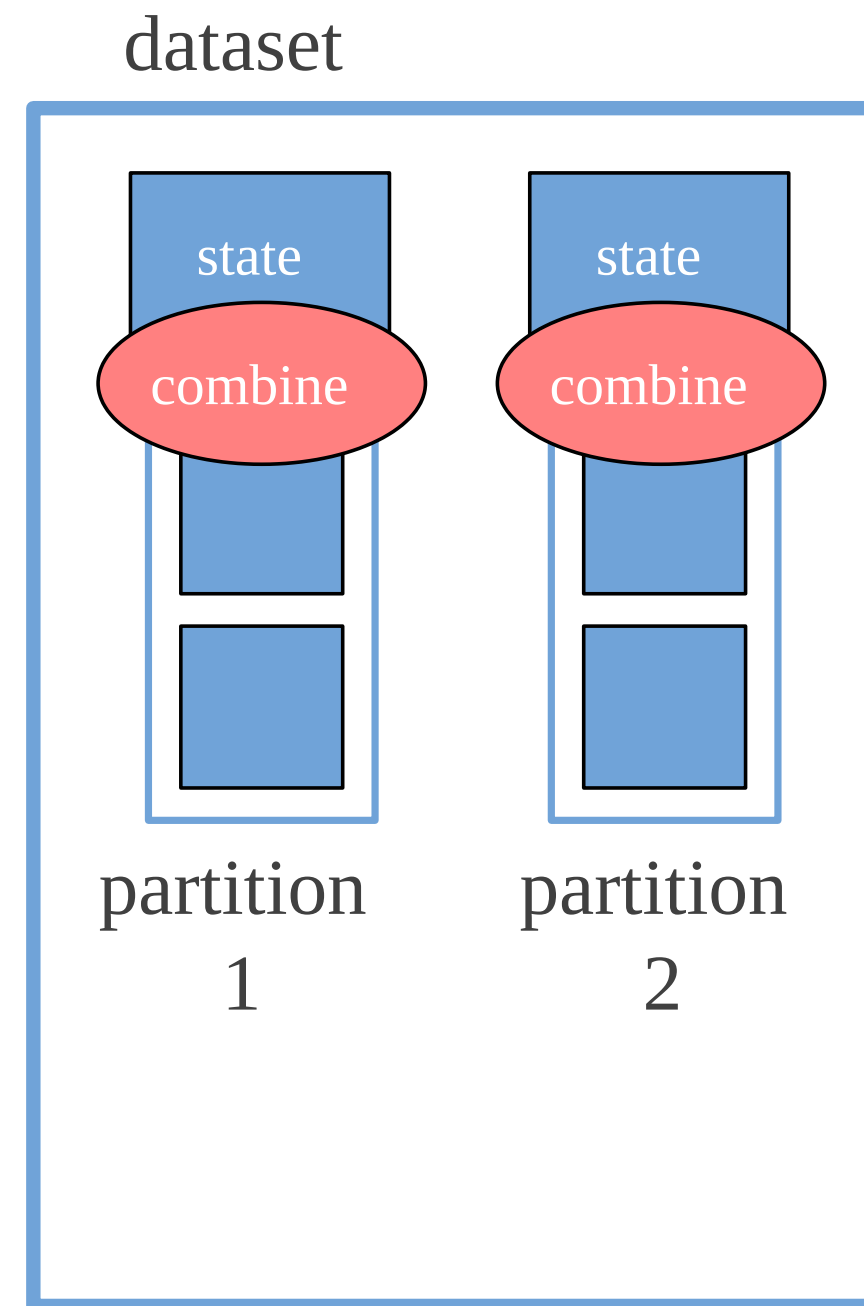


CPU 1

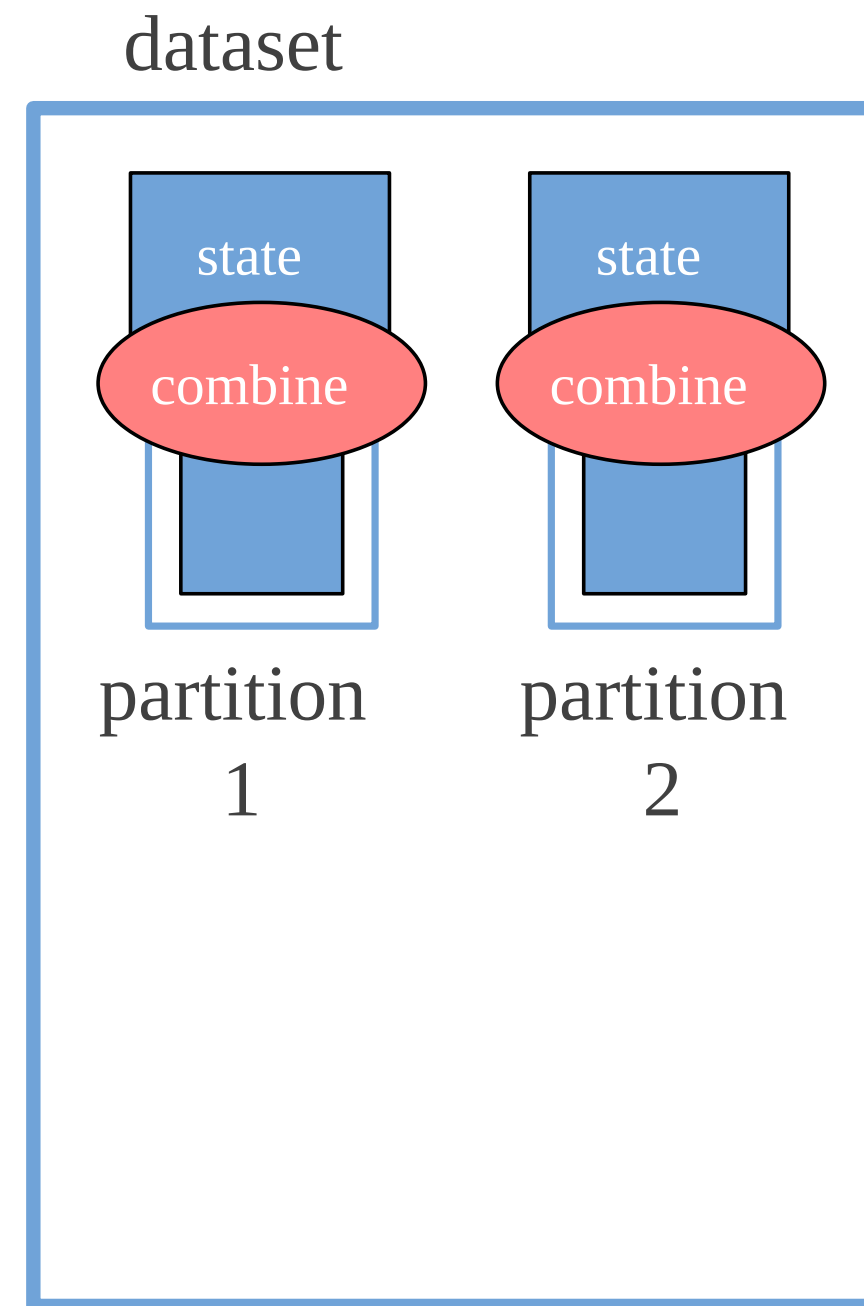
CPU 2

-

fold intermediate results in parallel



fold intermediate results in parallel

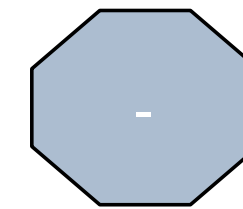
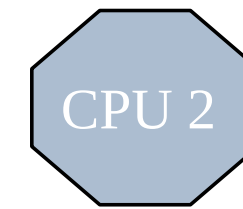
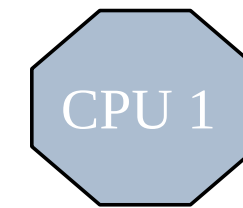
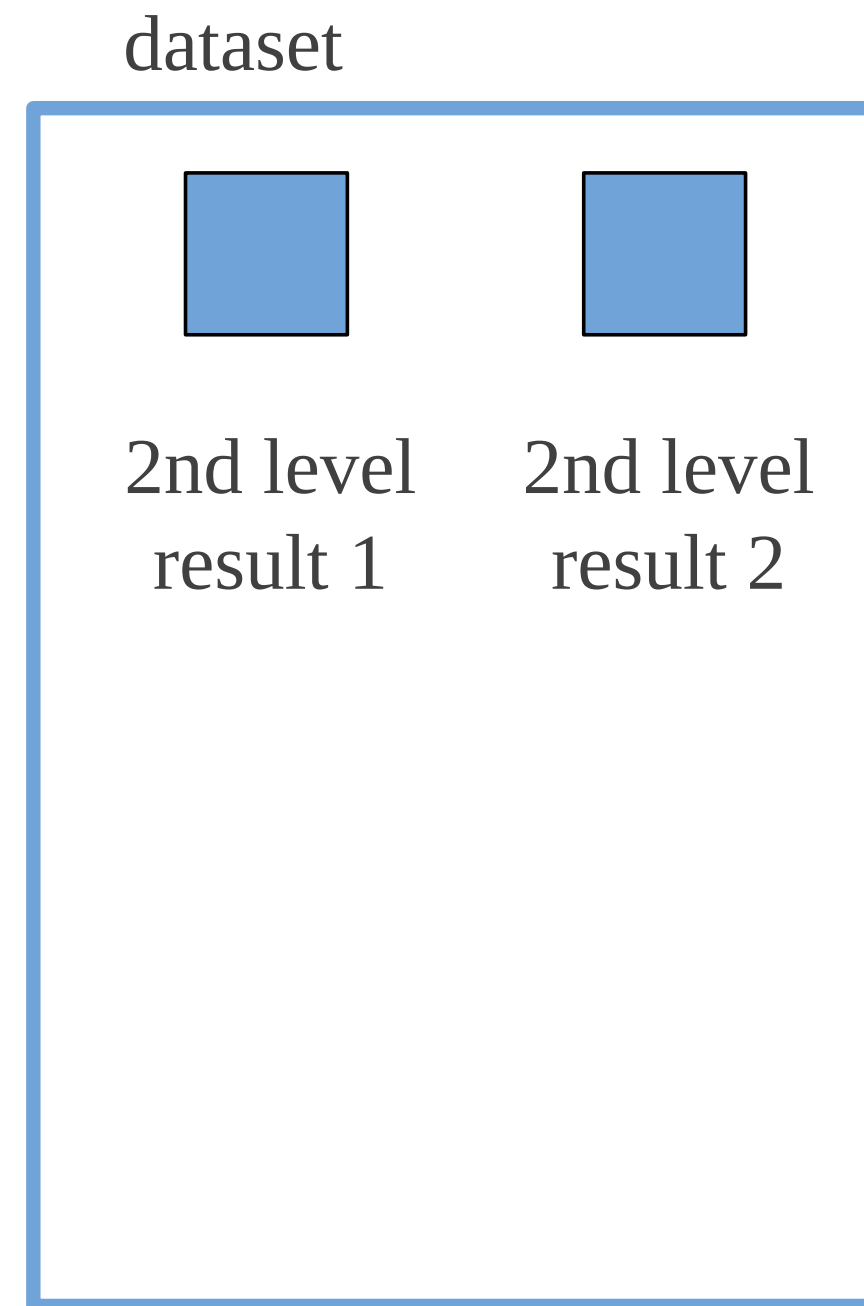


CPU 1

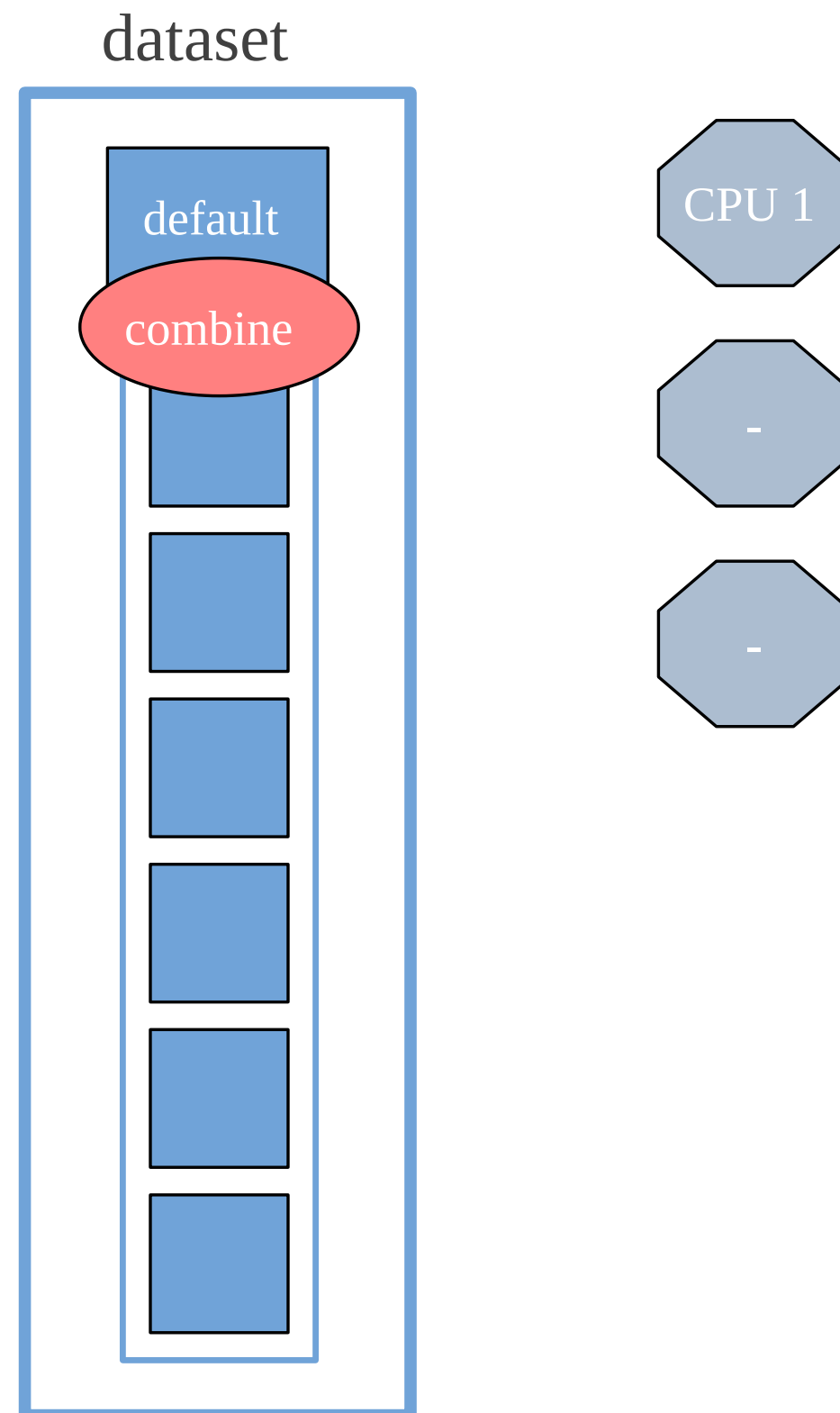
CPU 2

-

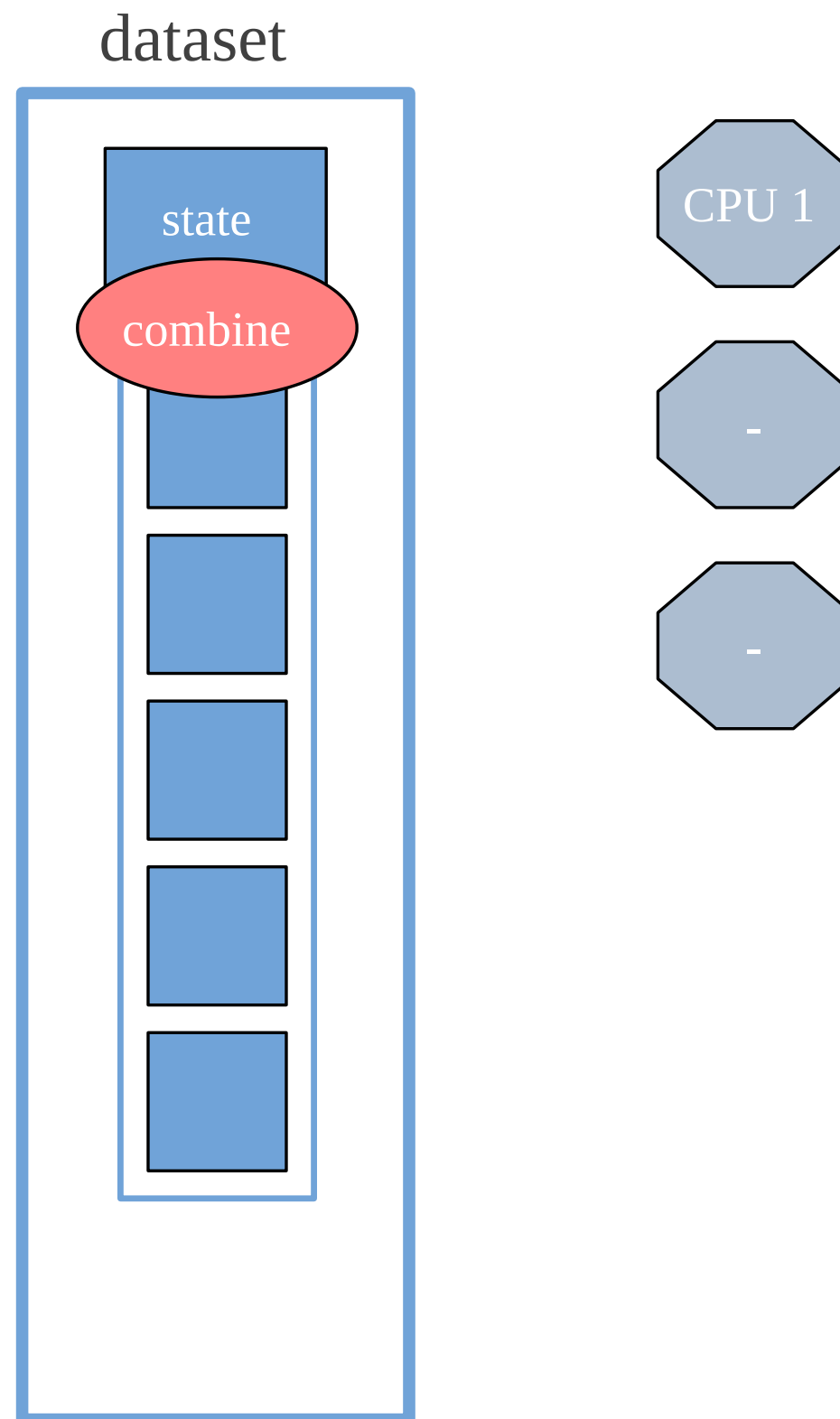
fold intermediate results in parallel



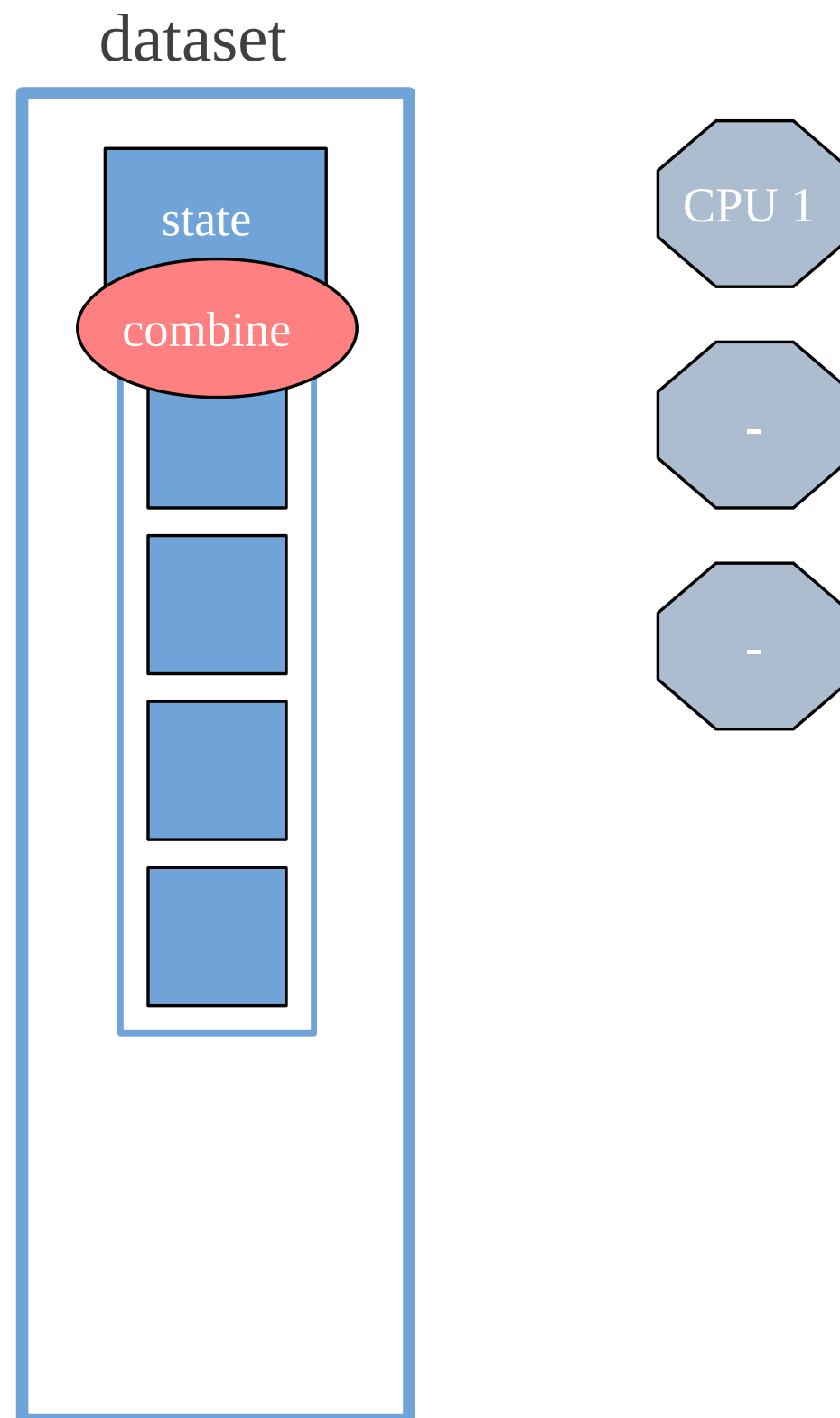
fold intermediate results sequentially



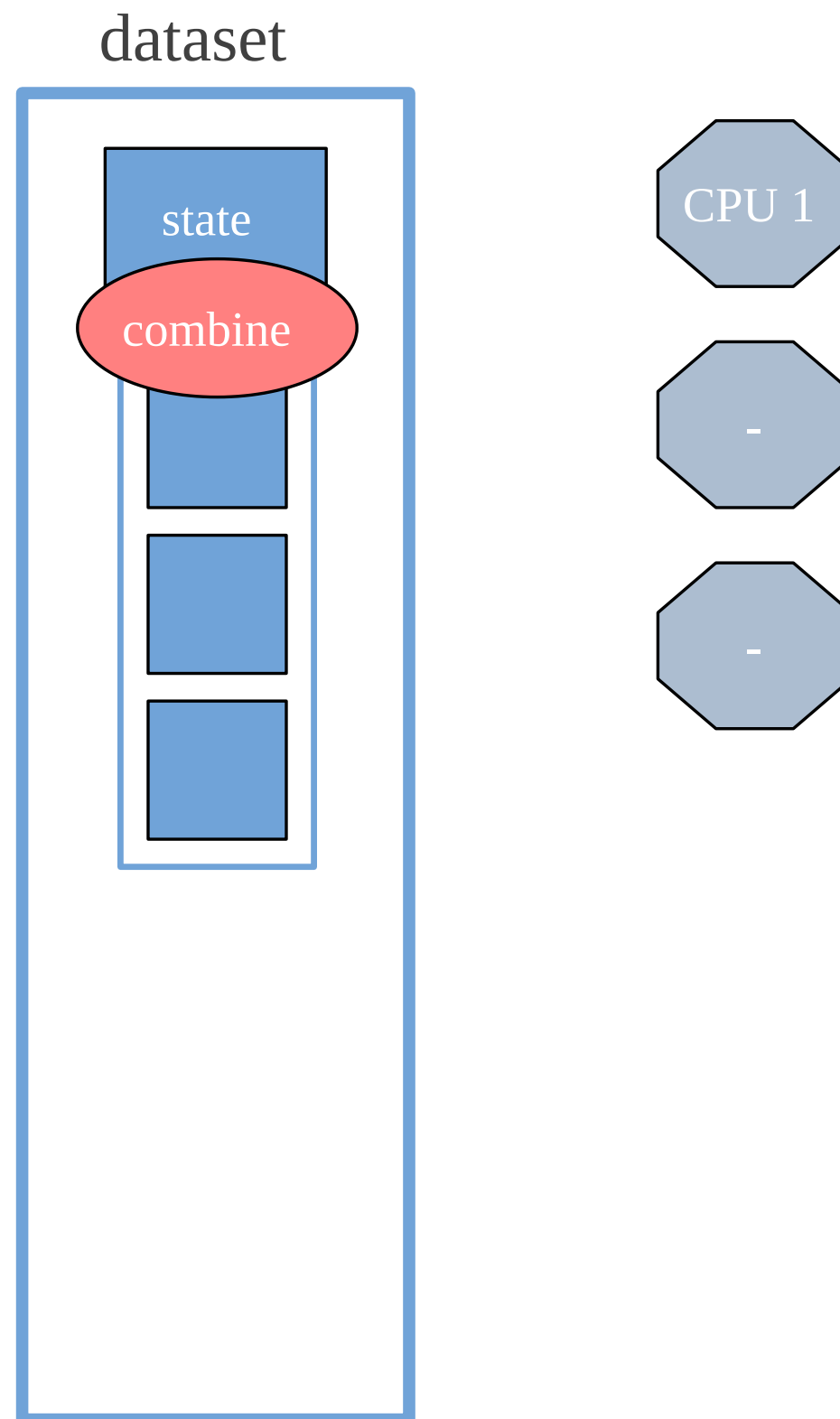
fold intermediate results sequentially



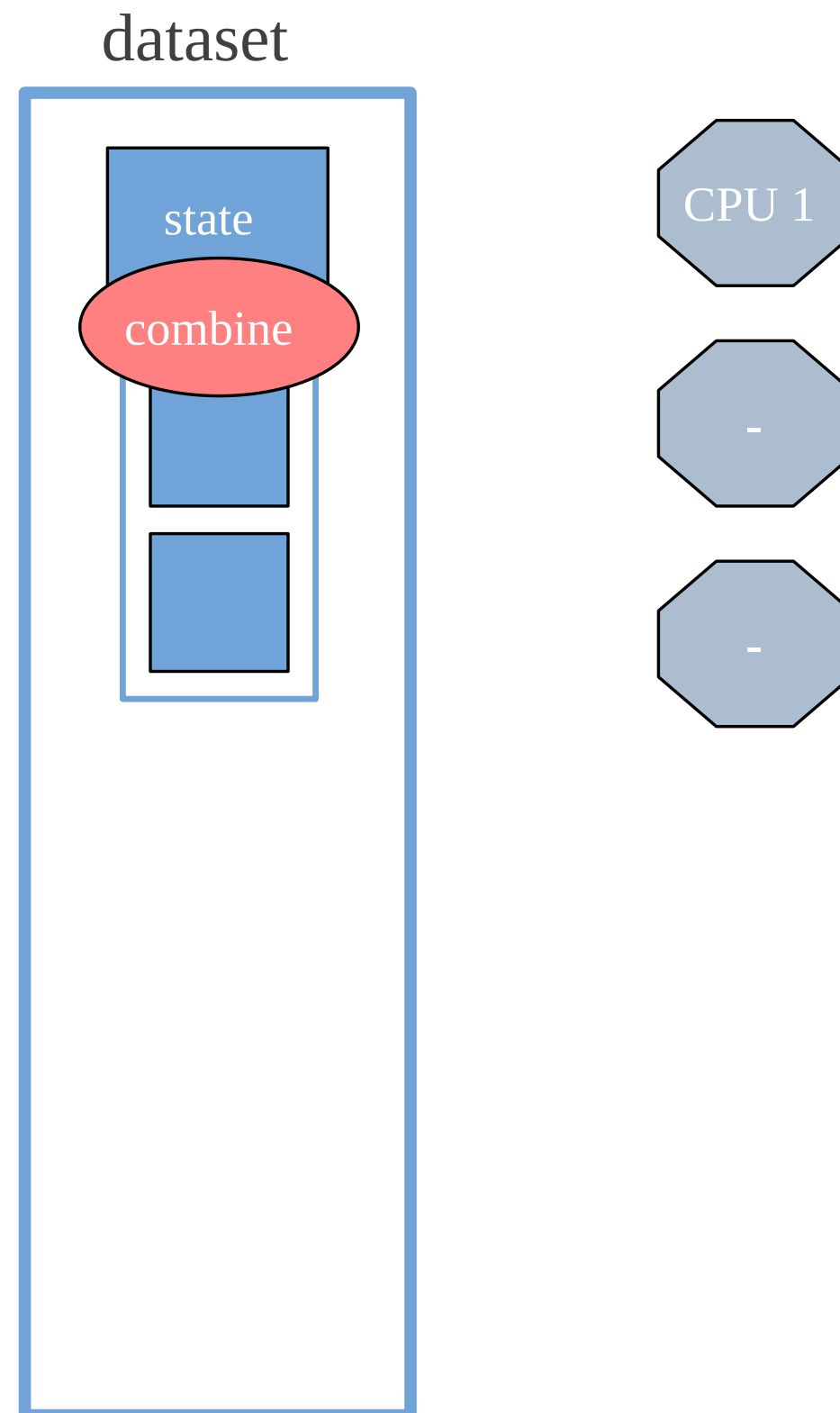
fold intermediate results sequentially



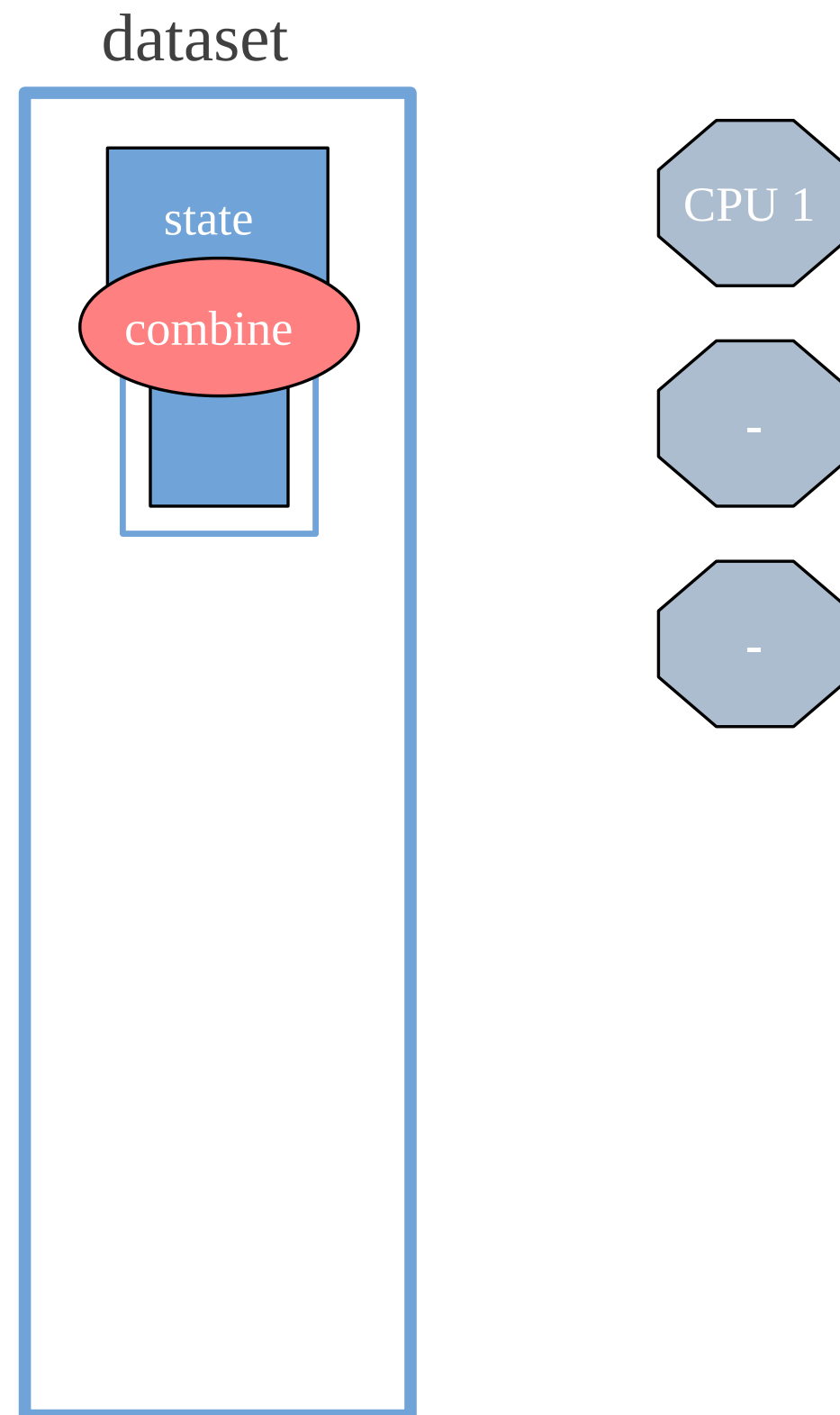
fold intermediate results sequentially



fold intermediate results sequentially

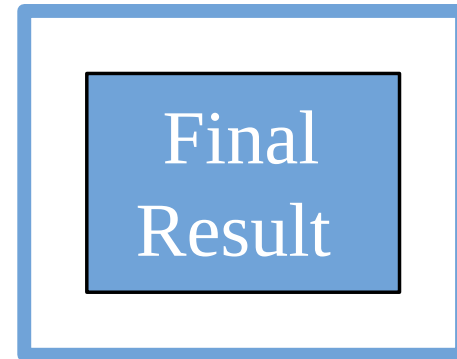


fold intermediate results sequentially



foldMap

dataset



Thread

```
def createThread(n: Int): Thread = new Thread {  
  override def run(): Unit =  
    println(s"Thread ${n}")  
}  
  
val threads = 1.to(4).map(createThread)
```

```
threads.foreach(_.start())  
// Thread 1  
// Thread 3  
// Thread 2  
// Thread 4
```

Executor and Runnable

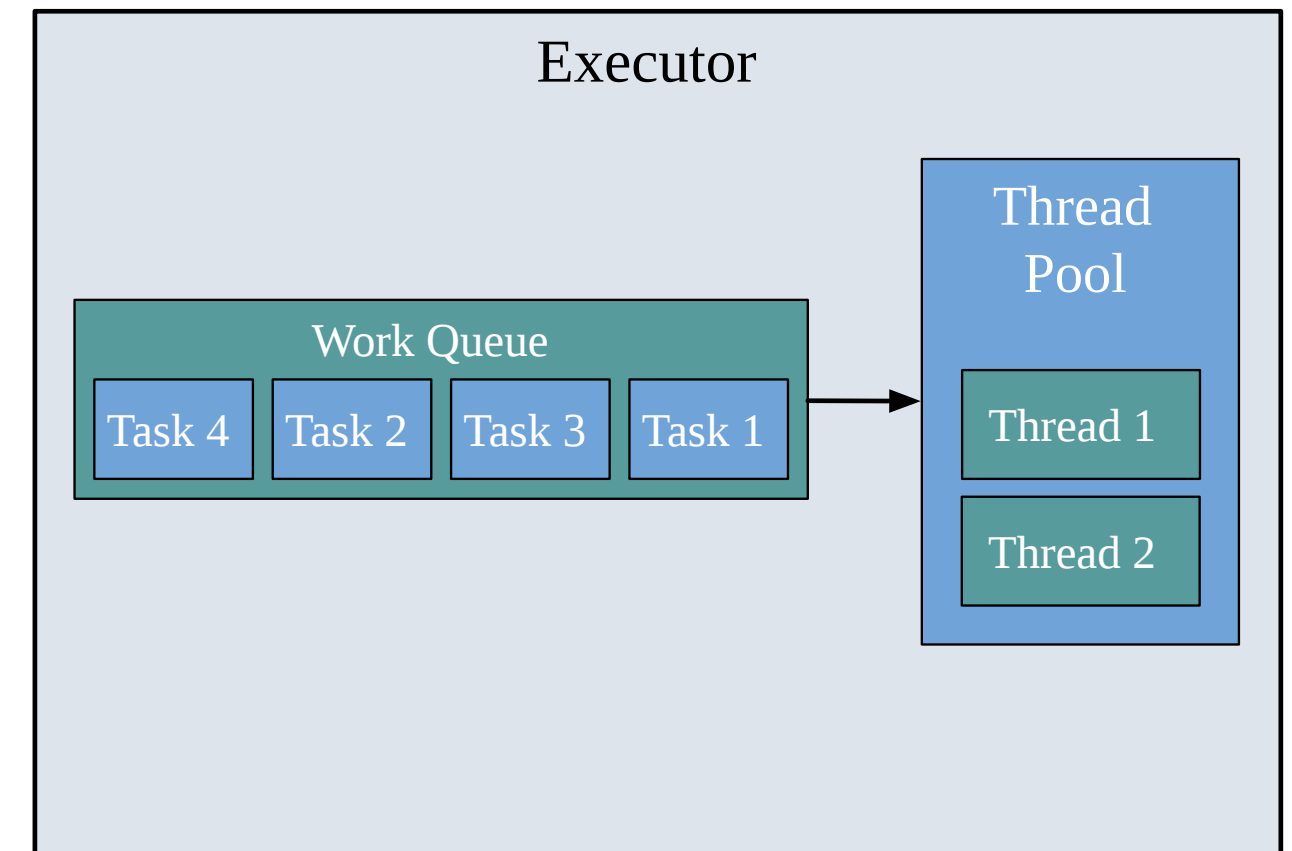
```
import java.util.concurrent.Executors

val fixedPool = Executors.newFixedThreadPool(2)

def createRunnable(n: Int): Runnable =
  new Runnable {
    def run(): Unit =
      println(s"Runnable ${n}")
  }

val runnables = 1.to(4).map(createRunnable)
```

```
runnables.foreach(fixedPool.submit)
// Runnable 1
// Runnable 3
// Runnable 2
// Runnable 4
```



Executor and Runnable

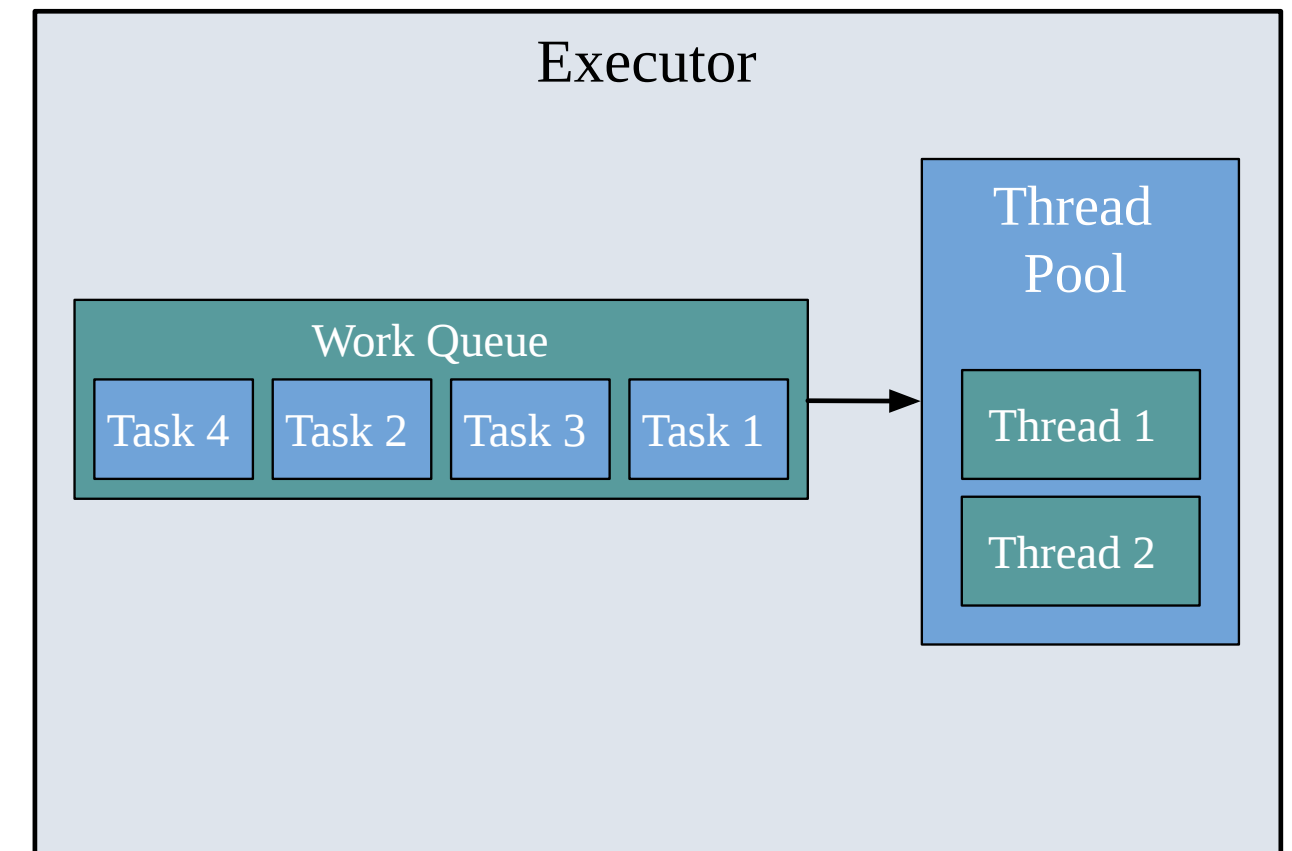
```
import java.util.concurrent.Executors

val fixedPool = Executors.newFixedThreadPool(2)

def createRunnable(n: Int): Runnable =
  new Runnable {
    def run(): Unit = {
      val thread = Thread.currentThread
      println(s"[${thread.getName}] Runnable ${n}")
    }
  }

val runnables = 1.to(4).map(createRunnable)
```

```
runnables.foreach(fixedPool.submit)
// [pool-19-thread-1] Runnable 1
// [pool-19-thread-2] Runnable 3
// [pool-19-thread-1] Runnable 2
// [pool-19-thread-2] Runnable 4
```



ExecutionContext and Future

```
import java.util.concurrent.Executors
import scala.concurrent.duration._
import scala.concurrent.{ Await, ExecutionContext, Future }

val fixedPool = Executors.newFixedThreadPool(2)
val executionContext = ExecutionContext.fromExecutor(fixedPool)
```

```
val future = Future {
  Thread.sleep(1000) // sleep 1 second
  1
}(executionContext)
// future: Future[Int] = Future(<not completed>)
```

ExecutionContext and Future

```
import java.util.concurrent.Executors
import scala.concurrent.duration._
import scala.concurrent.{ Await, ExecutionContext, Future }

val fixedPool = Executors.newFixedThreadPool(2)
implicit val executionContext = ExecutionContext.fromExecutor(fixedPool)
```

```
val task = Future {
  Thread.sleep(1000) // sleep 1 second
  3
}
// task: Future[Int] = Future(<not completed>)
```

ExecutionContext and Future

```
import java.util.concurrent.Executors
import scala.concurrent.duration._
import scala.concurrent.{ Await, ExecutionContext, Future }

val fixedPool = Executors.newFixedThreadPool(2)
implicit val executionContext = ExecutionContext.fromExecutor(fixedPool)
```

```
val task = Future {
  Thread.sleep(1000) // sleep 1 second
  3
}
// task: Future[Int] = Future(<not completed>)
```

```
Await.result(task, 2.minutes)
// res: Int = 3
```

ExecutionContext and Future

```
import java.util.concurrent.Executors
import scala.concurrent.duration._
import scala.concurrent.{ Await, ExecutionContext, Future }

val fixedPool = Executors.newFixedThreadPool(2)
implicit val executionContext = ExecutionContext.fromExecutor(fixedPool)
```

```
val task = Future {
  Thread.sleep(1000 * 60 * 5) // sleep 5 minutes
  3
}
// task: Future[Int] = Future(<not completed>)
```

```
Await.result(task, 2.minutes)
// java.util.concurrent.TimeoutException: Future timed out after [2 minutes]
```

ExecutionContext and Future

```
import java.util.concurrent.Executors
import scala.concurrent.duration._
import scala.concurrent.{ Await, ExecutionContext, Future }

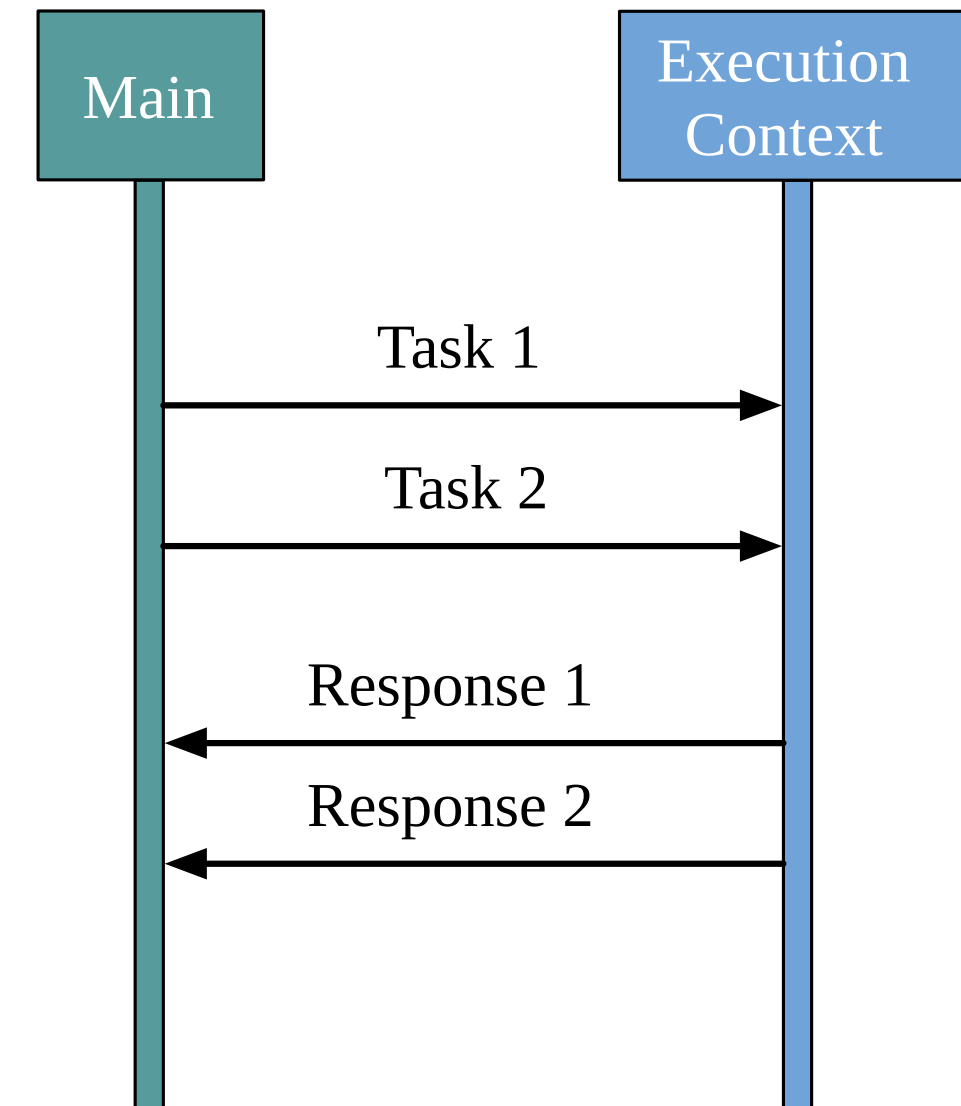
val fixedPool = Executors.newFixedThreadPool(2)
implicit val executionContext = ExecutionContext.fromExecutor(fixedPool)
```

```
val task = Future {
  Thread.sleep(1000 * 60 * 5) // sleep 5 minutes
  3
}
// task: Future[Int] = Future(<not completed>)
```

```
Await.result(task, Duration.Inf)
// res: Int = 3
```

ExecutionContext and Future

```
val future1 = Future { task(1) }  
val future2 = Future { task(2) }  
  
val response1 = Await.result(future1, Duration.Inf)  
val response2 = Await.result(future2, Duration.Inf)
```



ExecutionContext and Future

```
val future1    = Future { task(1) }  
val response1 = Await.result(future1, Duration.Inf)  
  
val future2    = Future { task(2) }  
val response2 = Await.result(future2, Duration.Inf)
```

