

remoteparfor>remoteparfor.getCompleteIntervals (Calls: 7214, Time: 29687.086 s)

Generated 14-Sep-2025 22:57:07 using performance time.  
Class method in file C:\Program Files\MATLAB\R2021a\toolbox\parallel\distcomp\+distcomp\remoteparfor.m  
[Copy to new window for comparing multiple runs](#)

Parents (calling functions)

Function Name	Function Type	Calls
<a href="#">parallel_function&gt;distributed_execution</a>	Subfunction	7214

Lines that take the most time

Line Number	Code	Calls	Total Time (s)	% Time	Time Plot
<a href="#">379</a>	<code>r = q.poll(1, timeUnitSeconds);</code>	42875	29647.781	99.9%	<div></div>
<a href="#">380</a>	<code>obj.displayOutput();</code>	42875	17.492	0.1%	
<a href="#">410</a>	<code>thisdata = obj.deserialize(data(2));</code>	19832	9.256	0.0%	
<a href="#">387</a>	<code>if ~obj.Session.isSessionRunning</code>	23043	3.904	0.0%	
<a href="#">409</a>	<code>data = r.getResult;</code>	19832	1.966	0.0%	
All other lines			6.686	0.0%	
Totals			29687.086	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time (s)	% Time	Time Plot
<a href="#">java.util.concurrent.LinkedBlockingQueue</a>	Java method	42875	29642.170	99.8%	<div></div>
<a href="#">remoteparfor&gt;remoteparfor.displayOutput</a>	Class method	42875	16.604	0.1%	
<a href="#">remoteparfor&gt;remoteparfor.deserialize</a>	Class method	19832	7.050	0.0%	
<a href="#">JavaBackedSession&gt;JavaBackedSession.isSessionRunning</a>	Class method	23043	3.288	0.0%	
<a href="#">...omp.pmode.parfor.ParforControllerImpl\$IntervalResultImpl</a>	Java method	59496	1.176	0.0%	
<a href="#">remoteparfor&gt;iTimeUnitSeconds</a>	Class method	7214	0.295	0.0%	
Self time (built-ins, overhead, etc.)			16.504	0.1%	
Totals			29687.086	100%	

Code Analyzer results

No Code Analyzer messages.

Coverage results

[Show coverage for parent folder](#)

Total lines in function	54
Non-code lines (comments, blank lines)	15
Code lines (lines that can run)	39
Code lines that did run	33
Code lines that did not run	6
Coverage (did run/can run)	84.62 %

Function listing

Time	Calls	Line	
		368	<code>function [tags, results] = getCompleteIntervals(obj, numIntervals)</code>
0.040	7214	<a href="#">369</a>	<code>q = obj.IntervalCompleteQueue;</code>
0.345	7214	<a href="#">370</a>	<code>timeUnitSeconds = <a href="#">iTimeUnitSeconds</a>();</code>
0.048	7214	<a href="#">371</a>	<code>tags = nan(numIntervals, 1);</code>
0.039	7214	<a href="#">372</a>	<code>results = cell(numIntervals, 2);</code>
0.004	7214	<a href="#">373</a>	<code>for i = 1:numIntervals</code>
0.072	19832	<a href="#">374</a>	<code>    r = [];</code>
0.005	19832	<a href="#">375</a>	<code>    err = [];</code>

```

0.058 42875 377      assert(obj.NumIntervalsInController > 0, ...
378      'Internal error in PARFOR - no intervals to retrieve.');
```

```

29647.781 42875 379      r = q.poll(1, TimeUnit.SECONDS);
```

```

17.492 42875 380      obj.displayOutput();
381      % In each poll tick of the polling wait, yield to any
382      % events allowed to interrupt the parallel language.
0.833 42875 383      parallel.internal.pool.yield();
0.592 42875 384      if isempty(r)
385          % Only test to see if the session is failing if we didn't get a
386          % results from the queue
3.904 23043 387          if ~obj.Session.isSessionRunning
388              errorMessageInput = iGetParpoolLinkForError();
389              error(message('parallel:lang:parfor:SessionShutDown', errorMessageInput));
0.006 23043 390          end
0.003 19832 391      else
0.062 19832 392          obj.NumIntervalsInController = obj.NumIntervalsInController - 1;
1.897 19832 393          if r.hasError
394              % Maybe try again
395              [r, err] = obj.handleIntervalErrorResult(r);
396              q = obj.IntervalCompleteQueue;
0.003 19832 397          end
0.010 42875 398      end
0.374 42875 399      end
400      % Check to see if the interval result has an error
0.004 19832 401      if ~isempty(err)
402          % Java code has already interrupted the remote execution of the parfor
403          % body and halted the receipt of further IO. Before throwing an error,
404          % we perform the normal cleanup activities.
405          obj.complete(false);
406          throw(err);
0.003 19832 407      else
1.289 19832 408          tags(i) = r.getTag;
1.966 19832 409          data = r.getResult;
9.256 19832 410          thisdata = obj.deserialize(data(2));
0.075 19832 411          assert(numel(thisdata) == 2, ...
412              'Unexpectedly received the incorrect number of outputs.');
```

```

0.199 19832 413          results(i,:) = thisdata;
0.478 19832 414          obj.IncompleteIntervalsCell{tags(i)} = [];
```

```

0.003 19832 415      end
0.006 19832 416      end
417      % Yield one final time to ensure all events directly generated by
418      % code in the PARFOR loop are executed before we give control back
419      % to the user.
0.095 7214 420      parallel.internal.pool.yield();
0.102 7214 421      end

```

Local functions in this file are not included in this listing.