spmd_numint.m

spmd - single program, multiple data

http://www.mathworks.com/examples/parallel-computing/400-using-gop-to-achieve-mpi_allreduce-functionality#4 /html

(More) References:

http://se.mathworks.com/help/distcomp/examples/numerical-estimation-of-pi-using-message-passing.html

doc spmd

http://se.mathworks.com/help/distcomp/introduction-to-parallel-solutions.html?refresh=true#brjw1fx-3

Some explanations taken from the references

- An spmd block runs on the workers of the existing parallel pool. If no pool exists, spmd will start a new parallel pool, unless the automatic starting of pools is disabled in your parallel preferences. If there is no parallel pool and spmd cannot start one, the code runs serially in the client session.
- The objects created are called composite objects, they "behave" like cell arrays.

```
AA{n} returns the values of AA from worker n.
```

```
AA(n) returns a container of the content of AA from worker n
```

Open pool.

p = gcp; numlabs=p.NumWorkers

Or if you want to set the value to numlabs:

```
nlabs=2; % laptop
%nlabs=8; % Triton
parpool(nlabs)
```

Numerical integration example

$$f(x) = \int_0^1 \frac{4}{1+x^2}\,dx = \arctan(1) = \pi$$

Function (handle) to integrate:

```
f=@(x) 4 ./(1+x.^2);
```

Define the variables a and b on all the workers, but let their values depend on labindex so that the intervals [a, b] correspond to the subintervals shown in the figure. We then verify that the intervals are correct. Note that the code in the body of the spmd statement is executed in parallel on all the workers in the parallel pool.

```
spmd
 a = (labindex - 1)/nlabs;
 b = labindex/nlabs;
fprintf('Subinterval: [%-4g, %-4g]\n', a, b)
end
```

```
Starting parallel pool (parpool) using the 'local' profile ... connected to 2 workers.
Lab 1:
  Subinterval: [0   , 0.5 ]
Lab 2:
  Subinterval: [0.5 , 1   ]
```

```
spmd
   myIntegral = integral(f, a, b)
end
```

```
Lab 1:

  myIntegral =

      1.8546

Lab 2:

  myIntegral =

      1.2870
```

```
myIntegral
```

```
myIntegral =

   Lab 1: class = double, size = [1  1]
   Lab 2: class = double, size = [1  1]
```

Sum myIntegral over all labs:

```
spmd
    % piApprox = gplus(myIntegral)
    piApprox = gplus(myIntegral,1)
end
```

```
Lab 1:

  piApprox =

      3.1416

Lab 2:

  piApprox =

       []
```

```
format long
piApprox{1}  % In the former case both (all) get the value
```

```
ans =
   3.141592653589793
```

```
piApprox{2}  % In the latter case this (the rest) get [].
```

```
ans =

     []
```

```
delete(gcp)
```

GOP - GlobalOPeration

<html>
http://www.mathworks.com/examples/parallel-computing/400-using-gop-to-achieve-mpi_allreduce-functionality#4
</html>

Exercises, developments:

- Integrate other functions on other intervals
- Use lower order methods (trapez, Simpson) on subintervals-> gain accuracy/efficiency from parallelism.