distributed

Create distributed array from data in the client workspace or a datastore

Syntax

```
D = distributed(ds)
D = distributed(X)
D = distributed(C,dim)
```

Description

`D = distributed(ds)` creates a distributed array from a `datastore` `ds`. `D` is a distributed array stored in parts on the workers of the open parallel pool. example

To retrieve the distributed array elements from the pool back to an array in the MATLAB® workspace, use `gather`.

`D = distributed(X)` creates a distributed array from an array `X`. example

Constructing a distributed array from local data this way is appropriate only if the MATLAB client can store the entirety of `X` in its memory. To construct large distributed arrays, use one of the constructor methods such as `ones(___,'distributed')`, `zeros(___,'distributed')`, etc. For a list, see Constructor.

If the input argument is already a distributed array, the result is the same as the input.

`D = distributed(C,dim)` creates a distributed array from a Composite array `C`, with the entries of `C` concatenated and distributed along the dimension `dim`. If you omit `dim`, then the first dimension is the distribution dimension. example

All entries of the Composite array must have the same class. Dimensions other than the distribution dimension must match.

Examples

Create Distributed Arrays

Create a small array and distribute it.

```
Nsmall = 50;
D1 = distributed(magic(Nsmall));
```

Create a large distributed array directly, using a build method.

```
Nlarge = 1000;
D2 = rand(Nlarge,'distributed');
```

Retrieve elements of a distributed array, and note where the arrays are located by their `Class`.

```
D3 = gather(D2);
whos
```

```
  Name           Size            Bytes  Class

  D1               50x50             733  distributed
  D2             1000x1000           733  distributed
  D3             1000x1000       8000000  double
  Nlarge           1x1                 8  double
  Nsmall           1x1                 8  double
```

Create a Distributed Array from a Datastore

This example shows how to create and load distributed arrays using `datastore`. You first create a datastore using an example data set. This data set is too small to show equal partitioning of the data over the workers. To simulate a large data set, artificially increase the size of the datastore using `repmat`.

```
files = repmat({'airlinesmall.csv'}, 10, 1);
ds = tabularTextDatastore(files);
```

Select the example variables.

```
ds.SelectedVariableNames = {'DepTime','DepDelay'};
ds.TreatAsMissing = 'NA';
```

Create a distributed table by reading the datastore in parallel. Partition the datastore with one partition per worker. Each worker then reads all data from the corresponding partition. The files must be in a shared location accessible from the workers.

```
dt = distributed(ds);
```

```
Starting parallel pool (parpool) using the 'local' profile ... connected to 4 workers.
```

Finally, display summary information about the distributed table.

```
summary(dt)
```

```
Variables:

    DepTime: 1,235,230×1 double
        Values:

            min           1
            max        2505
            NaNs     23,510

    DepDelay: 1,235,230×1 double
        Values:

            min       -1036
            max        1438
            NaNs     23,510
```

Create a Distributed Array from a Composite Array

Start a parallel pool of workers and create a `Composite` array by using `spmd`.

```
p = parpool("local",4);
```

```
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4).
```

```
spmd
C = rand(3,labindex-1);
end
C
```

```
C =

   Lab 1: class = double, size = [3  0]
   Lab 2: class = double, size = [3  1]
   Lab 3: class = double, size = [3  2]
   Lab 4: class = double, size = [3  3]
```

To create a distributed array out of the `Composite` array, use the `distributed` function. For this example, distribute the entries along the second dimension.

```
d = distributed(C,2)
```

```
d =

    0.6383    0.9730    0.2934    0.3241    0.9401    0.1897
    0.5195    0.7104    0.1558    0.0078    0.3231    0.3685
    0.1398    0.3614    0.3421    0.9383    0.3569    0.5250
```

```
spmd
    d
end
```

```
Lab 1:

  This worker does not store any elements of d.

Lab 2:

  This worker stores d(:,1).

         LocalPart: [3x1 double]
      Codistributor: [1x1 codistributor1d]

Lab 3:

  This worker stores d(:,2:3).

         LocalPart: [3x2 double]
      Codistributor: [1x1 codistributor1d]

Lab 4:

  This worker stores d(:,4:6).

         LocalPart: [3x3 double]
      Codistributor: [1x1 codistributor1d]
```

When you are finished with the computations, delete the parallel pool.

```
delete(p);
```

Tips

- A distributed array is created on the workers of the existing parallel pool. If no pool exists, `distributed` starts a new parallel pool unless the automatic starting of pools is disabled in your parallel preferences. If there is no parallel pool and `distributed` cannot start one, the result is the full array in the client workspace.

See Also

codistributed | datastore | gather | ones | parpool | spmd | tall | zeros