alex-konovalov Update README.md c900a8a a minute ago

1 contributor

80 lines (60 sloc) 3 KB Raw Blame History

Distributed calculations with the SCSCP package

This directory contains:

- `avgord.g` - file from the GAP Software Carpentry lesson
- `gapd.sh` - script to start one GAP SCSCP server
- `gapfarm.sh` - script to start a farm of GAP SCSCP servers
- `myserver.g` - configuration file for GAP SCSCP server
- `parsearch.g` - GAP code for the function ParSearchForGroupExamples to perform parallel search in the GAP Small Groups Library. It takes four arguments: the order to check, the number of the first and the last group and the chunksize. This file also sets InfoLevel and the list of SCSCP servers to use.

Setting up

1. Install GAP.
2. Check that the IO package is built: if `LoadPackage("io");` returns `fail`, you have to compile it. Otherwise the SCSCP package will not be loaded.
3. Edit the path to `bin/gap.sh` file in the line 51 of `gapd.sh`. If you need to specify any command line options for GAP SCSCP servers, do this here.
4. Leave as many calls to `gapd.sh` in the `gapfarm.sh` script as the number of cores on your computer. Remove or comment out other calls of `gapd.sh`
5. Update the line setting up `SCSCPservers` in `parsearch.g` making port numbers in the list `[ 26101 .. 26102 ]` matching those in `gapfarm.sh`.
6. Call `./gapfarm.sh` to start the "farm" of GAP SCSCP servers.
7. Start GAP with `gap avgord.g parsearch.g`
8. You should be able to call `ParSearchForGroupExamples` which takes four arguments: the order to check, the number of the first and the last group and the chunksize, for example:

```
gap> n:=96;ParSearchForGroupExamples(n,1,NrSmallGroups(n),30);
96
#I  1/8:master --> localhost:26101 : [ 96, 1, 30 ]
#I  2/8:master --> localhost:26102 : [ 96, 31, 60 ]
#I  localhost:26102 --> 2/8:master : [  ]
#I  3/8:master --> localhost:26102 : [ 96, 61, 90 ]
#I  localhost:26101 --> 1/8:master : [  ]
#I  4/8:master --> localhost:26101 : [ 96, 91, 120 ]
#I  localhost:26102 --> 3/8:master : [  ]
#I  5/8:master --> localhost:26102 : [ 96, 121, 150 ]
#I  localhost:26101 --> 4/8:master : [  ]
#I  6/8:master --> localhost:26101 : [ 96, 151, 180 ]
#I  localhost:26102 --> 5/8:master : [  ]
#I  7/8:master --> localhost:26102 : [ 96, 181, 210 ]
#I  localhost:26101 --> 6/8:master : [  ]
#I  8/8:master --> localhost:26101 : [ 96, 211, 231 ]
#I  localhost:26101 --> 8/8:master : [  ]
#I  localhost:26102 --> 7/8:master : [  ]
[ [  ], [  ], [  ], [  ], [  ], [  ], [  ], [  ] ]
```

And here is another group with integer average order of its elements:

```
gap> n:=1785;ParSearchForGroupExamples(n,1,NrSmallGroups(n),1);
1785
#I  1/2:master --> localhost:26101 : [ 1785, 1, 1 ]
#I  2/2:master --> localhost:26102 : [ 1785, 2, 2 ]
#I  localhost:26101 --> 1/2:master : [ 1785, 1 ]
#I  localhost:26102 --> 2/2:master : [  ]
[ [ 1785, 1 ], [  ] ]
```

(Of course, no parallelisation is needed to check groups of order 1785, but it is needed to check whether there are other groups with such property among those available in the Small Groups Library).