18.11.2017

created by rebirther

BEHIND SRBASE - A SHORT GUIDE

Work reservation and preparation

http://www.mersenneforum.org/forumdisplay.php?f=81

The link is used for any base reservation. This avoid any duplicate work from other users.

Some more infos:

http://www.noprimeleftbehind.net/crus/Riesel-conjecture-reserves.htm

http://www.noprimeleftbehind.net/crus/Sierp-conjecture-reserves.htm

All bases over 1M tests (lower ranges) will be split into pieces. Normally there are 2-3 batches, 25-50k, 50-60k, 60-100k. Most of the work is done in Excel.

Work creation

A sievefile for example with 500k tests will be split into 500k WUs by a splitting script.

Here is a snippet of a sievefile. The first line is the header file used for the llr app:

```
100000000000:P:1:955:257
2014 50000
6292 50000
15406 50000
17910 50000
24376 50000
34942 50000
47458 50000
```

splitting script (split a sievefile line by line)

```php
<?php
$time_start = microtime(true);
$file = "R253";
$outputpath = "Input11";
// PRP File einlesen, ignoriert leere Zeilen und entfernt die Zeilenumbrüche
$data = @file($file, FILE_SKIP_EMPTY_LINES+FILE_IGNORE_NEW_LINES);
//$zeilen = count($data);
$zeilen = 1538;
echo "Zeilen: ". $zeilen."\r\n";
//print_r($data);
//mkdir($outputpath, 0777);
$i = 1;
$j=4763;
do {
    // String auf 7 Stellen auffüllen
    $stellen = sprintf("%07s", $j);
    $output = fopen("$outputpath/input11a22_$stellen.prp","x");
    if($output === FALSE) {
        echo "Fehler beim erstellen von input_$stellen.prp. Datei bereits vorhanden.\r\n";
        exit();
    }
    // Immer die erste Zeile in die Datei schreiben, Windowszeilenumbruch anfügen
    fwrite($output, $data[0]."\r\n");
    // jeweilige ausgelesene Zeile in die Datei schreiben, Windowszeilenumbruch anfügen
    fwrite($output, $data[$i]."\r\n");
    fclose($output);
    // echo "Datei ". $i ." erstellt.<br />";
    $i++;
    $j++;
} while ($i < $zeilen);

echo "Alle Dateien erstellt.\n";
// Script Execution Time
$time_end = microtime(true);
$time = $time_end - $time_start;
//echo "Process Time: {$time}";

function convert($size)
{
    $unit=array('b','kb','mb','gb','tb','pb');
    return @round($size/pow(1024,($i=floor(log($size,1024)))),2).' '.$unit[$i];
}
```

make_work script (work creation for BOINC)

```bash
#!/bin/bash

#FILES=/home/boincadm/projects/sr5/Input9/*
FILES=./Input9/*

let count=5856

for f in $FILES
do
  ((count++))
  echo $count

  # $f enthält den gesamten Pfad zur Datei (absolut oder relativ)
  # deshalb wird hier der Dateiname abgespalten und in $input gespeichert
  input=$(basename $f)

  # stage_file benötigt den kompletten Pfad zur Datei deshalb $f
  #cmd="./bin/stage_file --copy $f"
  # das folgende Kommando verschiebt die Datei anstatt zu kopieren
  cmd="./bin/stage_file $f"
  echo $cmd
  $cmd

  # create_work braucht nur den Namen der input Datei, der Pfad ist ja intern schon bekannt
  cmd2="./bin/create_work --appname srbase9 \
        --wu_name R648_350-400k_wu_${count} \
        --wu_template templates/srb_in.xml \
        --result_template templates/srb_out.xml ${input}"
  echo $cmd2
  $cmd2

done
```

input template (is used for a batch of WUs)

```
<input_template>
<file_info>
<number>0</number>
</file_info>
<workunit>
<file_ref>
<file_number>0</file_number>
<open_name>input.prp</open_name>
<copy_file/>
</file_ref>
  <min_quorum>1</min_quorum>
  <target_nresults>1</target_nresults>
  <max_error_results>10</max_error_results>
  <max_total_results>10</max_total_results>
  <max_success_results>2</max_success_results>
  <rsc_fpops_bound>8.2548e25</rsc_fpops_bound>
  <rsc_fpops_est>8e10</rsc_fpops_est>
  <delay_bound>172800</delay_bound>
  <credit>0.6</credit>
  <command_line>-d -oPgenInputFile=input.prp -oPgenOutputFile=primes.txt -oDiskWriteTime=10 -oOutputIterations=50000
oResultsFileIterations=99999999</command_line>
</workunit>
</input_template>
```

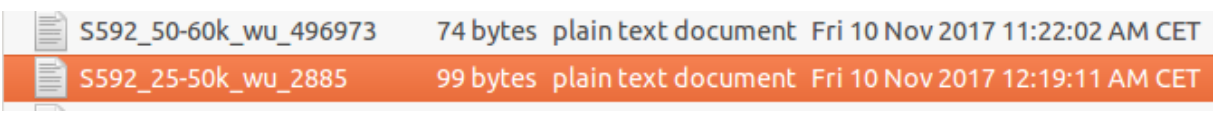
output_template (define lresults.txt as output file)

```
<output_template>
  <file_info>
    <name><OUTFILE_0/></name>
    <generated_locally/>
    <upload_when_present/>
    <max_nbytes>5000000</max_nbytes>
    <url><UPLOAD_URL/></url>
  </file_info>
  <result>
    <file_ref>
      <file_name><OUTFILE_0/></file_name>
      <open_name>lresults.txt</open_name>
      <copy_file/>
    </file_ref>
  </result>
</output_template>
```

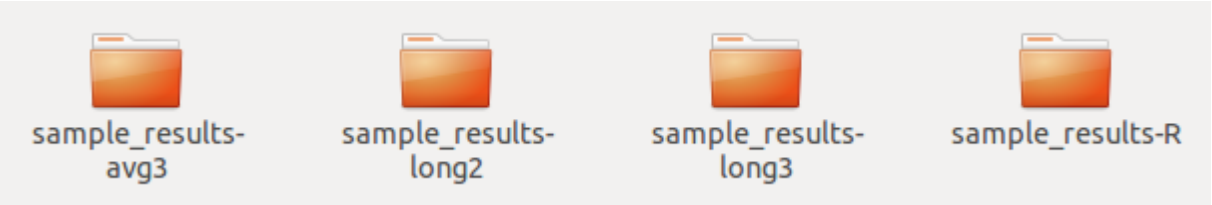
Checking results

There are some app issues with the output of the stderr.txt. On Mac and Linux there is no output in this file. Thats why the result files are very important. Any interruption of network or a VM/Server crash making these files useless (no output). Before these files are written the database entry is the first part.

There is a difference in these 2 files:
The first one is prime, the second not. All files over 110 byte need to be checked carefully.



Some output folders created by the assimilator (app by app):



merge.php (merging all results from a folder to lresults.txt)

```php
<?php
$resultfile = "lresults.txt";
$dir = opendir("R3");
        while ($file = readdir($dir)) {
            if ($file != "." && $file != "..") {
              // Open
              $fh = fopen("R3/". $file, "r");
              // Einlesen
              $content = fread($fh, filesize("R3/". $file));
              // Close
              fclose($fh);
              // Inhalt auslesen
              $content = file_get_contents("R3/". $file);
              // Output Open
              $output = fopen("$resultfile","a");
              // In Datei Schreiben, Zeilenumbruch am Ende der gleich wieder entfernt wird
              fwrite($output, str_replace("<br />","",nl2br("$content")));
              // Output Close
              fclose($output);
              }
         }
    closeDir($dir);
?>
```

Removing primes from a sievefile with the srfile app:

```
srfile_win64 -d primes.txt t17_b63-full.prp -G

pause
```

Calculation of runtime

sample:

Starting N+1 prime test of 828*1017^222660-1
Using FFT length 320K, Pass1=320, Pass2=1K, a = 3

828*1017^222660-1, bit: 50000 / 2224407 [2.24%]. Time per bit: 13.409 ms.
828*1017^222660-1, bit: 100000 / 2224407 [4.49%]. Time per bit: 12.986 ms.

formula:

Timesteps * time per bit / 1000 /60s =estimated runtime (the 2nd entry is important)