Redis thread implementation in PHP without compilation on CentOS

Redis is a useful solution when design systems on server and required the minimum latency between the client and server for handling message. We can easily build a thread application if we use shell_exec based on & and wait. PHP also provides a solution named as pthreads. but unfortunately the pthreads extension cannot be used in a web server environment. Threading in PHP should therefore remain to CLI-based applications only.

To know more about thread application based on the shell script, please refer http://qsok.com/display/KB/Thread+implementation+in+shell+script for more information.

To implement, thread-based redis operation, we need below two functions:

```
function redis_set($host,$port,$key,$value,$thread=false)
{
        $obj = [ "v" => $value ];
        $value = str_replace( "\"", "\\\"", json_encode( $obj, JSON_UNESCAPED_UNICODE));
        $cmd="redis-cli -h {$host} -p {$port} set \"{$key}\" \"{$value}\"";
        if ($thread==true) $cmd .= " &";

        $resp=trim( shell_exec( $cmd));

        if ($resp=="OK") return NULL;
        else
        {
                echo "{$resp}\n";
                return $resp;
        }
}

function redis_get($host,$port,$key)
{
        $cmd = "redis-cli -h {$host} -p {$port} get \"{$key}\"";
        $resp=trim(shell_exec( $cmd));
        $obj=json_decode($resp,JSON_UNESCAPED_UNICODE);
        return $obj['v'];
}
```

Once you call redis_set() in thread mode, you can wait until your process' done as following:

```
shell_exec( "wait");
```

Below is a practical example to implement Redis application comparing the normal and thread processing :

```
<?php

$prev_time = 0;
function tick()
{
        global $prev_time;

        $cur_time = microtime(true);

        if ($prev_time) $diff = $cur_time - $prev_time;
        else $diff = "";

        echo $cur_time . " (" . $diff . ")\n";

        $prev_time = $cur_time;
}

function redis_set($host,$port,$key,$value,$run_in_thread=false)
{
        $obj = [ "v" => $value ];
        $value = str_replace( "\"", "\\\"", json_encode( $obj, JSON_UNESCAPED_UNICODE));
        $cmd="redis-cli -h {$host} -p {$port} set \"{$key}\" \"{$value}\"";
        if ($run_in_thread==true) $cmd .= " &";
```

```
        $resp=trim( shell_exec( $cmd));

        if ($resp=="OK") return NULL;
        else
        {
                echo "{$resp}\n";
                return $resp;
        }
}

function redis_get($host,$port,$key)
{
        $cmd = "redis-cli -h {$host} -p {$port} get \"{$key}\"";
        $resp=trim(shell_exec( $cmd));
        $obj=json_decode($resp,JSON_UNESCAPED_UNICODE);
        return $obj['v'];
}

echo "Error testing\n";
echo redis_set( "10.0.1.10", 6379, "chun", "kang1");
echo redis_get( "10.0.1.10", 6379, "chun"). "\n";

echo "Test Redis call in Normal mode!!\n";
$prev_time = 0;
tick();
redis_set( "10.0.1.11", 6379, "chun", "kang1");
redis_set( "10.0.1.12", 6379, "chun", "kang2");
redis_set( "10.0.1.13", 6379, "chun", "kang3");
tick();
echo redis_get( "10.0.1.11", 6379, "chun"). "\n";
echo redis_get( "10.0.1.12", 6379, "chun"). "\n";
echo redis_get( "10.0.1.13", 6379, "chun"). "\n";
tick();

echo "\n";
echo "Test same things in Thread mode\n";

$prev_time = 0;
tick();
redis_set( "10.0.1.11", 6379, "chun", "thread-kang1", true);
redis_set( "10.0.1.12", 6379, "chun", "thread-kang2", true);
redis_set( "10.0.1.13", 6379, "chun", "thread-kang3", true);
tick();

shell_exec("wait"); // wait until all workers to complete the job

echo redis_get( "10.0.1.11", 6379, "chun"). "\n";
echo redis_get( "10.0.1.12", 6379, "chun"). "\n";
echo redis_get( "10.0.1.13", 6379, "chun"). "\n";
tick();

?>
```

And below is the result of above code:

```
Error testing
Could not connect to Redis at 10.0.1.10:6379: No route to host

Could not connect to Redis at 10.0.1.10:6379: No route to host

Test Redis call in Normal mode!!
1622086912.1606 ()
1622086912.1754 (0.014813184738159)
kang1
kang2
kang3
1622086912.1895 (0.014098882675171)

Test same things in Thread mode
1622086912.1895 ()
1622086912.2041 (0.014595985412598)
thread-kang1
thread-kang2
thread-kang3
1622086912.2201 (0.015931129455566)
```

The benefit of above example is to enhance its processing performance, so you can do more for your purpose.