

GAUSS Mathematical and Statistical System™ 9.0

New features and enhancements:

New Functions:

- ◆ ThreadStat
- ◆ ThreadBegin
- ◆ ThreadEnd
- ◆ ThreadJoin

Enhancements

- ◆ Increased array support

Available Platforms:

32-bit: Windows, Linux

64-bit: Linux

Additional Platforms to Come:

32-bit: Mac OS X, HP UX11

64-bit: Windows AMD, Windows Itanium 2, Mac OS X, Solaris



Divide And Conquer With Multi-Threading!

Harness the power and speed of multi-core, multi-processor, and hyper-threaded systems with GAUSS™ 9.0!

GAUSS 9.0 now includes functions for multi-threading your programs, as well as increased array support in more intrinsics and operators. The new threading functions let you define independent sections of your program that will run at the same time. These threads share the same workspace and can access all of the same symbols, procedures and keywords. You can create as many threads as you want, allowing you to take advantage of as many processors as you have.

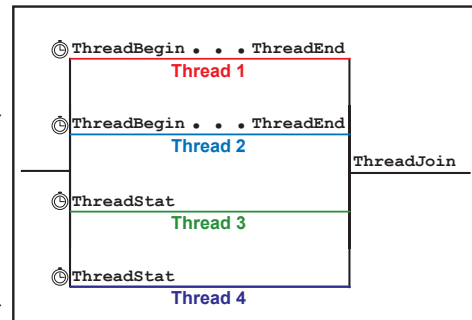
All threads in a set run simultaneously. You can create sets of threads anywhere in a program, including inside procedures and keywords. You can also create threads inside other threads, so you can multi-thread any portion of your programs or libraries that you want.

GAUSS 9.0 also features increased array support, allowing you to seamlessly use arrays in more of the operators and intrinsic functions.

Product Details

Using threads in your code allows you to take better advantage of the available processors on your machine. Dividing your code into multiple threads that run simultaneously can reduce the overall processing time of your programs.

To the right is an illustration that shows one thread set with four threads. Each thread processes at the same time as the other threads. Your program waits at the ThreadJoin command for all threads to finish. When the threads have completed, your program continues, making use of the work the threads have done. In the example here, this block of code could potentially run nearly four times faster on a quad core machine because the threads are running simultaneously.



for MORE information, contact :

RITME Informatique
34, boulevard Haussmann - 75009 Paris
tel : 01 42 46 00 42 - fax : 01 42 46 00 33
www.ritme.com - info@ritme.com

Threads can be created anywhere—in the main code, in procedures, in keywords. You can also create threads within other threads. This means you can multi-thread nearly anything you want and call it from anywhere in your program. You can multi-thread some or all of the procedures and keywords in your libraries and call them freely anywhere in your multi-threaded programs.

New Features and Enhancements

ThreadStat

The ThreadStat command allows you to set off a single statement as an individual thread. Usually this would be a line that takes some time to run and can be run simultaneously with other threads.

Example:

```
ThreadStat n = m'm;
```

ThreadBegin, ThreadEnd

The ThreadBegin and ThreadEnd commands are used to define a multi-line block of code that will be executed as a thread. ThreadBegin marks the beginning of the block, and ThreadEnd marks the end. The following is an example of the use of these commands:

```
ThreadBegin;
  y = x'x;
  z = y'y;
ThreadEnd;
```

ThreadJoin

ThreadJoin follows the final ThreadEnd or ThreadStat command in a set of threads. Your program waits at the ThreadJoin command for all threads in the preceding set to complete and then continues on. For example:

```
ThreadBegin;                                // Thread 1
  y = x'x;
  z = y'y;
ThreadEnd;
```

```
ThreadBegin;                                // Thread 2
  q = r'r;
  r = q'q;
ThreadEnd;
ThreadStat n = m'm;                          // Thread 3
ThreadStat p = o'o;                          // Thread 4
ThreadJoin;                                  // Program waits for
                                              // threads to complete

b = z + r + n'p;
```

The following code shows both a single-threaded and multi-threaded calculation of a moment:

```
proc mloop(x,N,its,pos); // Procedure
  local m;

  m = 0;
  for i(1,its,1);
    m = m + moment(x[pos:pos+N-1,],0);
    pos = pos + N;
  endfor;

  retp(m);
endp;

N = 4000;
x = rndn(N*N,4);
```

// SINGLE-THREADED CALCULATION

```
mlp = mloop(x, N, N, 1);
```

// MULTI-THREADED CALCULATION (split into four threads for a quad-core machine)

```
R = rows(x)/4;
ThreadStat m1 = mloop(x, N, N/4, 1);
ThreadStat m2 = mloop(x, N, N/4, 1+R);
ThreadStat m3 = mloop(x, N, N/4, 1+R*2);
ThreadStat m4 = mloop(x, N, N/4, 1+R*3);
ThreadJoin;
mtmlp = m1 + m2 + m3 + m4;
```

In time trials on a quad-core machine, the multi-threaded calculation ran 3.3 times faster than the single-threaded call!

Increased array support

GAUSS 9.0 now includes additional array support in more intrinsics and operators. More mathematical functions will handle arrays.

for MORE information, contact :

RITME Informatique
 34, boulevard Haussmann - 75009 Paris
 tel : 01 42 46 00 42 - fax : 01 42 46 00 33
 www.ritme.com - info@ritme.com