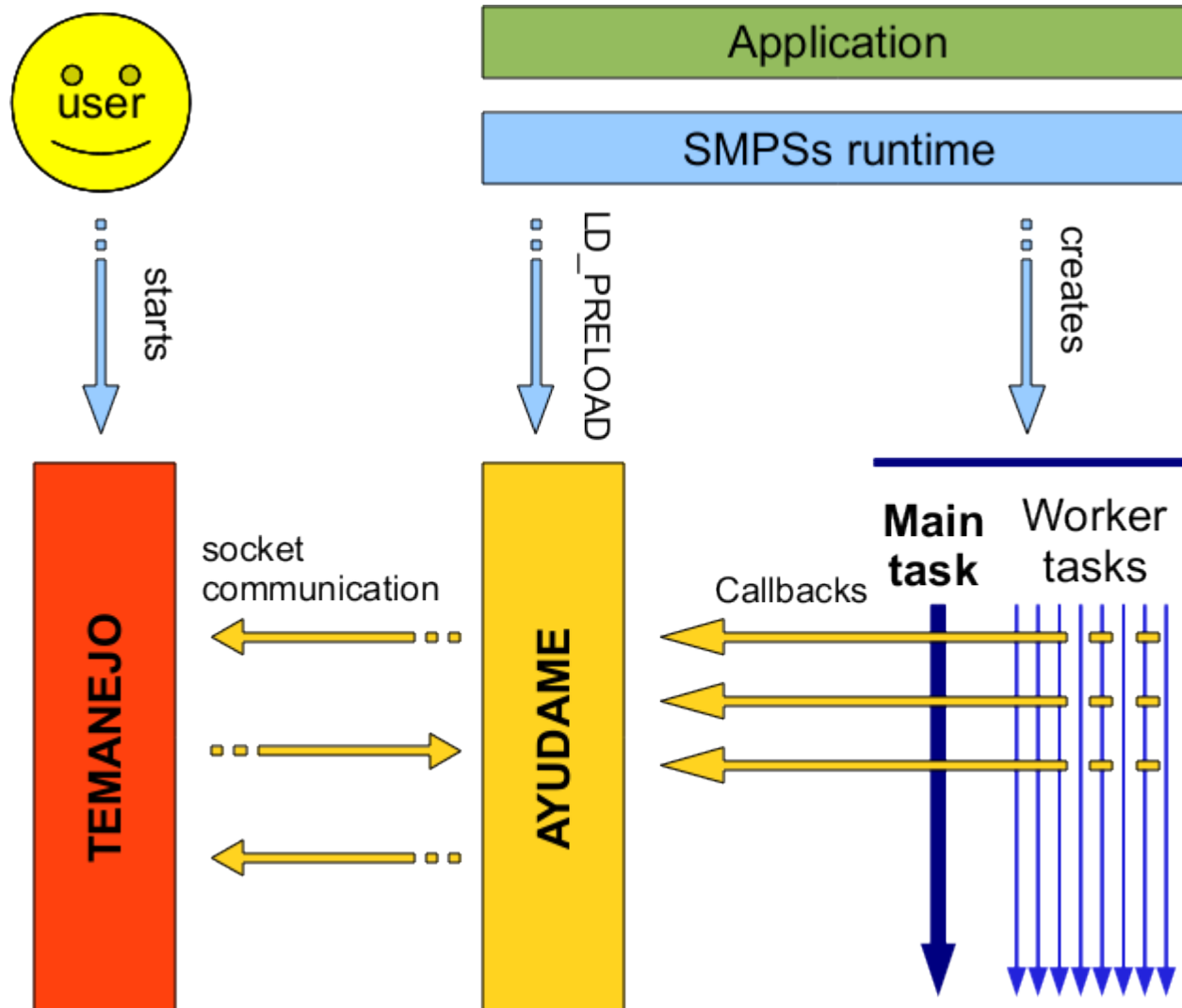


Introduction to the AYUDAME/TEMANEJO toolset

© HLRS, University of Stuttgart

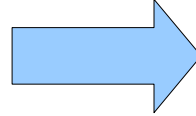
author: Steffen Brinkmann

Mode of operation



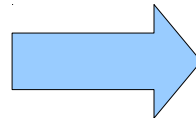
AYUDAME events

- AYU_PREINIT
- AYU_REGISTERTASK,
- AYU_INIT



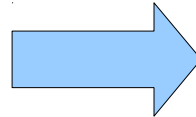
Initialisation: setting up socket, passing function names etc.

- AYU_ADDTASK
- AYU_ADDDEPENDENCY



build dependency graph

- AYU_PRESELECTTASK



force/block execution of task

- AYU_ADDTASKTOQUEUE
- AYU_PRERUNTASK
- (AYU_POSTRUNTASK)
- AYU_RUNTASKFAILED
- AYU_REMOVETASK



status of task

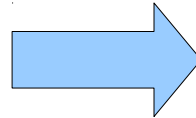
- AYU_WAITON
- AYU_BARRIER
- AYU_FINISH



breakpoints / cleanup

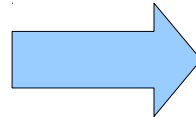
AYUDAME requests

- AYU_PAUSEONEVENT
- AYU_PAUSEONTASK,
- AYU_PAUSEONFUNCTION



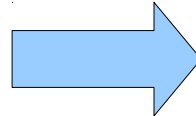
requests for pausing when a certain event happens/task is reached/function is entered

- AYU_STEP



allow to run until next pause/breakpoint

- AYU_BREAKPOINT



request to inhibit scheduling

Installation process

- Checkout new version from svn
- Prerequisites: NetworkX, python-gtk, python 2.4
- in SMPSSs dir:
 - autoreconf
 - ./configure --with-flavour=smp --enable-ayudame
 - make && make install
- in Tools/Ayudame dir:
 - make → libayudame.so
- compile your program:
 - smpss-cc -o your_app your_app.c
- in Tools/Ayudame dir:
 - ./temanejo /path/to/your_app

Startup

The screenshot displays the TEMANEJO application window. The title bar reads 'TEMANEJO'. The menu bar includes 'Connect', 'Preferences', 'Zoom In', 'Zoom Out', 'Best Fit', 'Normal Size', 'Run', 'Run 10 steps', 'Debug', 'Export', 'About', 'Help', and 'Quit'. The main toolbar contains icons for these actions. On the left, there is a table with task status and a legend for task states.

function	NQ	Q	R	F	total
total	0	0	0	0	0

Legend:

- ☐ NQ: not queued
- ☐ Q: queued
- ☐ R: running
- ☐ F: finished

The 'Connect to AYUDAME...' dialog box is open, showing the 'local connection' tab. It contains the following fields and buttons:

- 'Specify the executable:' with text './_jacobi' and an 'Open' button.
- 'Specify command line options for executable:' with an empty text field.
- 'Specify libayudame.so:' with text '/home/sbrinkma/git_clone/hlrs-text/Tools/Ayudame/libayudame.so' and an 'Open' button.
- 'Specify SMPSS configuration file:' with text '#enter configuration file here#' and an 'Open' button.
- 'Specify port:' with text '7756'.
- 'Specify number of CPUs:' with text '2'.
- 'Cancel' and 'Connect' buttons at the bottom right.

At the bottom of the main window, there is a status bar with the text 'No last message' and a timestamp 'Start session, Tue May 17 18:22:36 2011'.

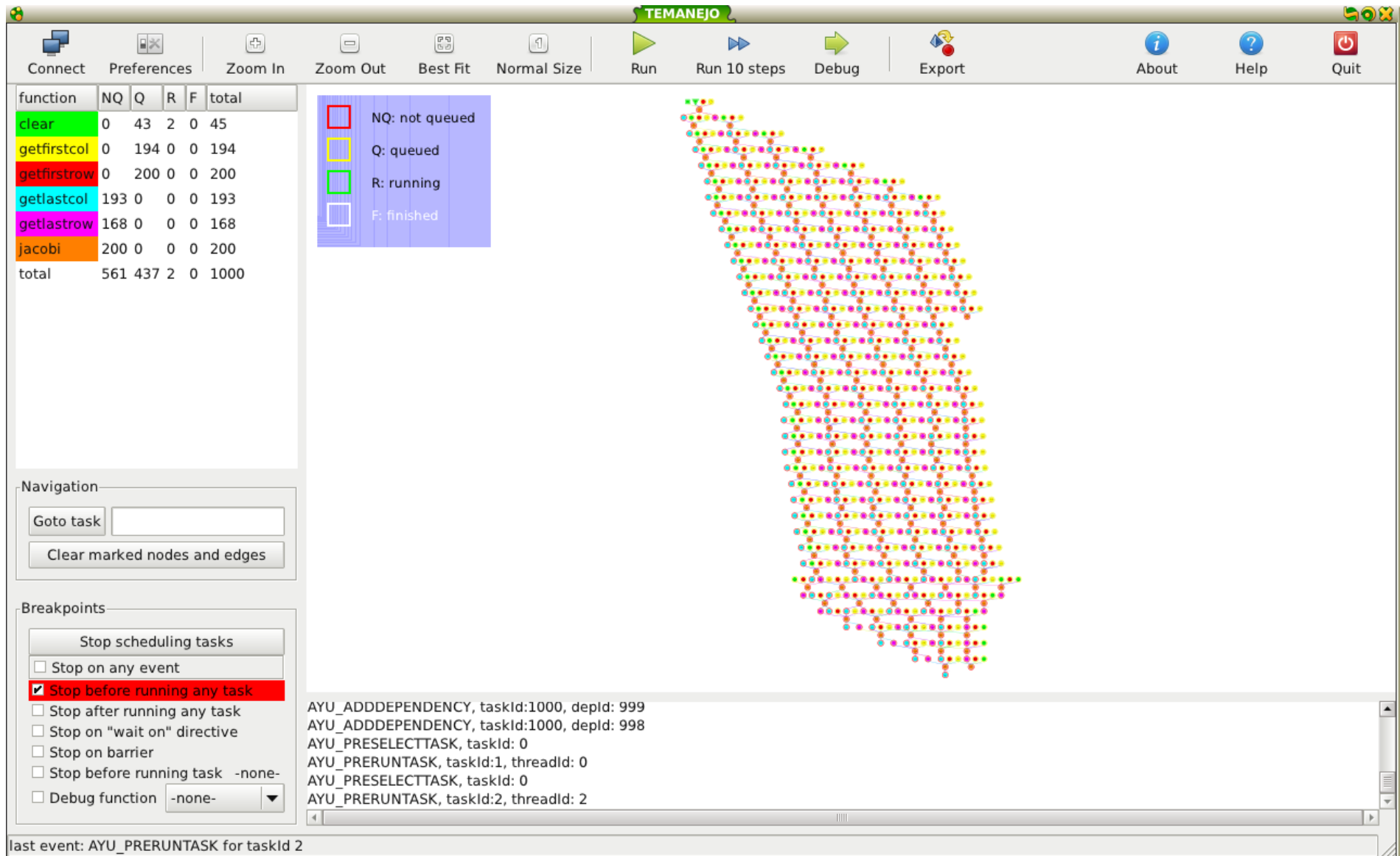
After pressing “connect”

The screenshot shows the TEMANEJO application window. The title bar is green and labeled 'TEMANEJO'. The menu bar includes 'Connect', 'Preferences', 'Zoom In', 'Zoom Out', 'Best Fit', 'Normal Size', 'Run', 'Run 10 steps', 'Debug', 'Export', 'About', 'Help', and 'Quit'. The main window is divided into several sections:

- Function Table:** A table with columns 'function', 'NQ', 'Q', 'R', 'F', and 'total'. The rows are 'clear' (green), 'getfirstcol' (yellow), 'getfirstrow' (red), 'getlastcol' (cyan), 'getlastrow' (magenta), 'jacobi' (orange), and 'total'.
- Legend:** A blue box containing four items: 'NQ: not queued' (red square), 'Q: queued' (yellow square), 'R: running' (green square), and 'F: finished' (blue square).
- Navigation:** A section with a 'Goto task' input field and a 'Clear marked nodes and edges' button.
- Breakpoints:** A section with a 'Stop scheduling tasks' button, a 'Stop on any event' checkbox, and a 'Stop before running any task' checkbox (checked). Below these are several other checkboxes for stopping on specific events or tasks.
- Task List:** A list of tasks at the bottom, including 'AYU_REGISTERTASK, taskid 0, funcld: 1 function name: getfirstcol', 'AYU_REGISTERTASK, taskid 0, funcld: 2 function name: getfirstrow', 'AYU_REGISTERTASK, taskid 0, funcld: 3 function name: getlastcol', 'AYU_REGISTERTASK, taskid 0, funcld: 4 function name: getlastrow', 'AYU_REGISTERTASK, taskid 0, funcld: 5 function name: jacobi', and 'AYU_INIT, taskid: 0'.

The status bar at the bottom left shows 'last event: AYU_INIT for taskid 0'.

After pressing “run”



Zooming in

The screenshot displays the TEMANEJO application interface. The main window shows a task dependency graph with nodes representing tasks and edges representing dependencies. The nodes are color-coded: red for 'N: not queued', yellow for 'Q: queued', green for 'R: running', and blue for 'F: finished'. The graph shows a complex network of tasks, with some nodes highlighted in red and others in yellow. The edges are labeled with memory addresses.

Navigation:

- Goto task:
- Clear marked nodes and edges

Breakpoints:

- Stop scheduling tasks
- ☐ Stop on any event
- ☒ Stop before running any task
- ☐ Stop after running any task
- ☐ Stop on "wait on" directive
- ☐ Stop on barrier
- ☐ Stop before running task -none-
- ☐ Debug function -none-

Task List:

function	NQ	Q	R	F	total
clear	0	43	2	0	45
getfirstcol	0	194	0	0	194
getfirstrow	0	200	0	0	200
getlastcol	193	0	0	0	193
getlastrow	168	0	0	0	168
jacobi	200	0	0	0	200
total	561	437	2	0	1000

Task Log:

- AYU_ADDDEPENDENCY, taskId:1000, depld: 999
- AYU_ADDDEPENDENCY, taskId:1000, depld: 998
- AYU_PRESELECTTASK, taskId: 0
- AYU_PRERUNTASK, taskId:1, threadId: 0
- AYU_PRESELECTTASK, taskId: 0
- AYU_PRERUNTASK, taskId:2, threadId: 2

last event: AYU_PRERUNTASK for taskId 2

Goto task 1

TEMANEJO

Connect Preferences Zoom In Zoom Out Best Fit Normal Size Run Run 10 steps Debug Export About Help Quit

function	NQ	Q	R	F	total
clear	0	43	2	0	45
getfirstcol	0	194	0	0	194
getfirstrow	0	200	0	0	200
getlastcol	193	0	0	0	193
getlastrow	168	0	0	0	168
jacobi	200	0	0	0	200
total	561	437	2	0	1000

Legend:

- NQ: not queued
- Q: queued
- R: running
- F: finished

Navigation:

Goto task Clear marked nodes and edges

Breakpoints:

Stop scheduling tasks

☐ Stop on any event

☒ Stop before running any task

☐ Stop after running any task

☐ Stop on "wait on" directive

☐ Stop on barrier

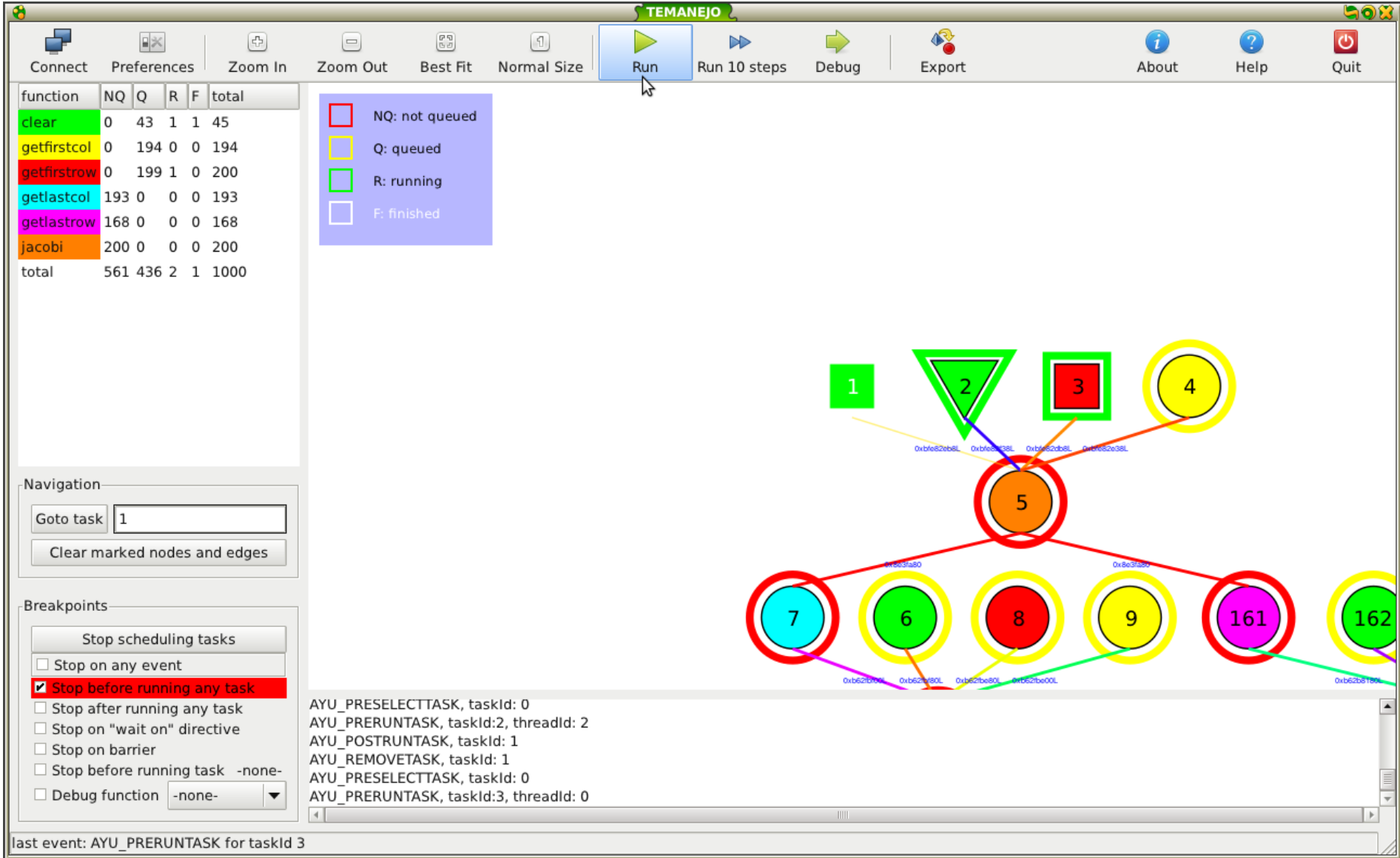
☐ Stop before running task -none-

☐ Debug function -none-

last event: AYU_PRERUNTASK for taskId 2

```
graph TD; 1[1] -- 0xb622e38L --> 5((5)); 2[2] -- 0xb622e38L --> 5; 3((3)) -- 0xb622e38L --> 5; 4((4)) -- 0xb622e38L --> 5; 5((5)) -- 0xb622e38L --> 7((7)); 5 -- 0xb622e38L --> 6((6)); 5 -- 0xb622e38L --> 8((8)); 5 -- 0xb622e38L --> 9((9)); 5 -- 0xb622e38L --> 161((161)); 5 -- 0xb622e38L --> 162((162));
```

Press “run”



Change node size and margin thickness in the preferences

The screenshot shows the TEMANEJO application interface. The main window displays a task dependency graph with nodes and edges. A legend on the left indicates the status of tasks: NQ (not queued), Q (queued), R (running), and F (finished). The Settings dialog box is open, showing the 'User interface' tab. The 'Margin thickness' is set to 15, and the 'Size of nodes' is set to 10. The 'Graph direction' is set to 'Top to bottom'. The 'Show key' checkbox is checked, and the 'Autozoom' checkbox is unchecked. The 'Run n steps' is set to 10. The 'Colourful dependencies' checkbox is checked, and the 'Simple display' checkbox is unchecked. The 'Cancel' and 'OK' buttons are visible at the bottom of the dialog box.

function NQ Q R F total

clear	0	43	1	1	45
getfirstcol	0	194	0	0	194
getfirstrow	0	199	1	0	200
getlastcol	193	0	0	0	193
getlastrow	168	0	0	0	168
jacobi	200	0	0	0	200
total	561	436	2	1	1000

Navigation

Goto task 1

Clear marked nodes and edges

Breakpoints

Stop scheduling tasks

☐ Stop on any event

☒ Stop before running any task

☐ Stop after running any task

☐ Stop on "wait on" directive

☐ Stop on barrier

☐ Stop before running task -none-

☐ Debug function -none-

last event: AYU_PRERUNTASK for taskId 3

AYU_PRESELECTTASK, taskId: 0

AYU_PRERUNTASK, taskId:2, threadId: 2

AYU_POSTRUNTASK, taskId: 1

AYU_REMOVETASK, taskId: 1

AYU_PRESELECTTASK, taskId: 0

AYU_PRERUNTASK, taskId:3, threadId: 0

Settings

User interface

☒ Show key

☐ Autozoom

Run n steps 10

☒ Colourful dependencies

☐ Simple display

Margin thickness 15

Size of nodes 10

Graph direction

☒ Top to bottom

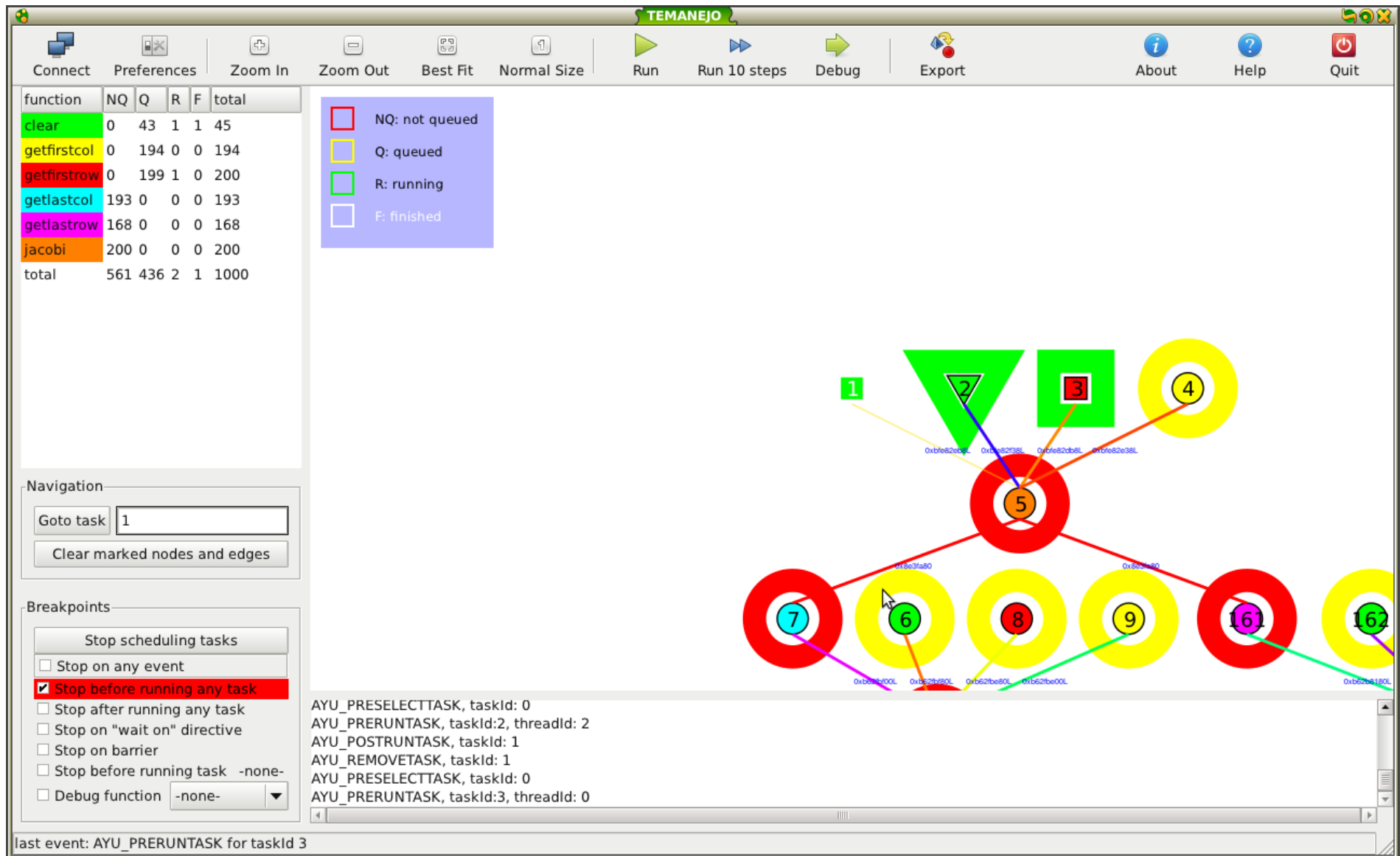
☐ Left to right

☐ Bottom to top

☐ Right to left

Cancel OK

ok...



Press “Best Fit”

The screenshot shows the TEMANEJO software interface. The top menu bar includes options like Connect, Preferences, Zoom In, Zoom Out, Best Fit (highlighted), Normal Size, Run, Run 10 steps, Debug, Export, About, Help, and Quit. A tooltip for 'Best Fit' says 'Zoom to fit'.

On the left, there is a table with task statistics:

function	NQ	Q	R	F	total
clear	0	43	1	1	45
getfirstcol	0	194	0	0	194
getfirstrow	0	199	1	0	200
getlastcol	193	0	0	0	193
getlastrow	168	0	0	0	168
jacobi	200	0	0	0	200
total	561	436	2	1	1000

Below the table is a 'Navigation' section with a 'Goto task' input field containing '1' and a 'Clear marked nodes and edges' button. Below that is a 'Breakpoints' section with several options, including 'Stop before running any task' which is checked.

The main area displays a large, complex graph of nodes and edges, representing task execution. A tooltip 'Tabelle einfügen' is visible over the graph.

At the bottom, a status bar shows the last event: 'last event: AYU_PRERUNTASK for taskId 3'.

Press “Run 10 steps” several times

The screenshot displays the TEMANEJO software interface, which is used for visualizing and controlling task execution. The main window shows a complex graph of tasks and their dependencies, represented by colored nodes (red, yellow, green, blue) and edges. The graph is titled "Run 10 steps" and includes a tooltip that reads: "Run program and ignore several pause events. Change the number of events in the preferences."

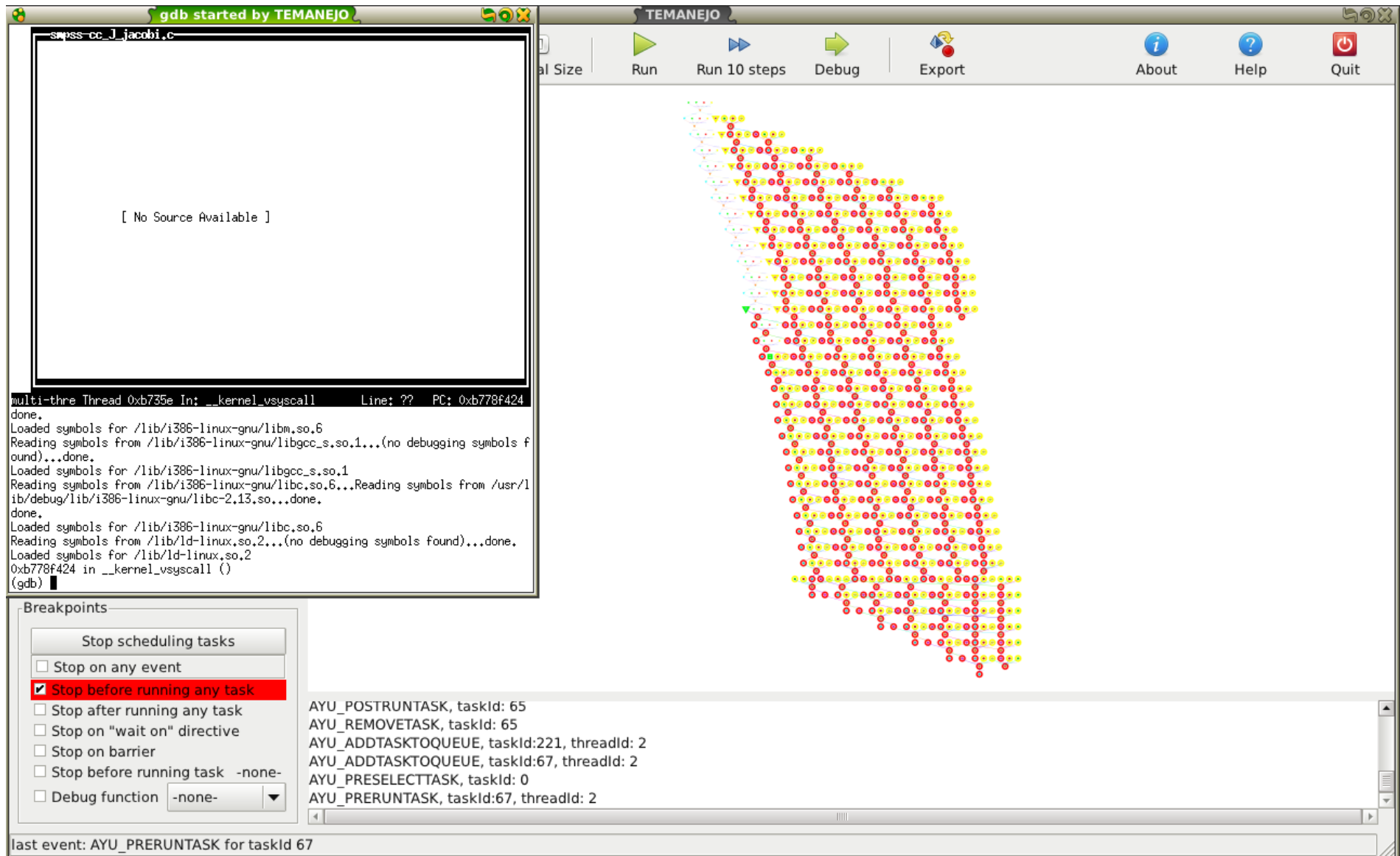
On the left side, there is a table showing the execution status of various functions:

function	NQ	Q	R	F	total
clear	0	28	1	16	45
getfirstcol	0	179	0	15	194
getfirstrow	0	185	0	15	200
getlastcol	184	0	1	8	193
getlastrow	159	1	0	8	168
jacobi	191	0	0	9	200
total	534	393	2	71	1000

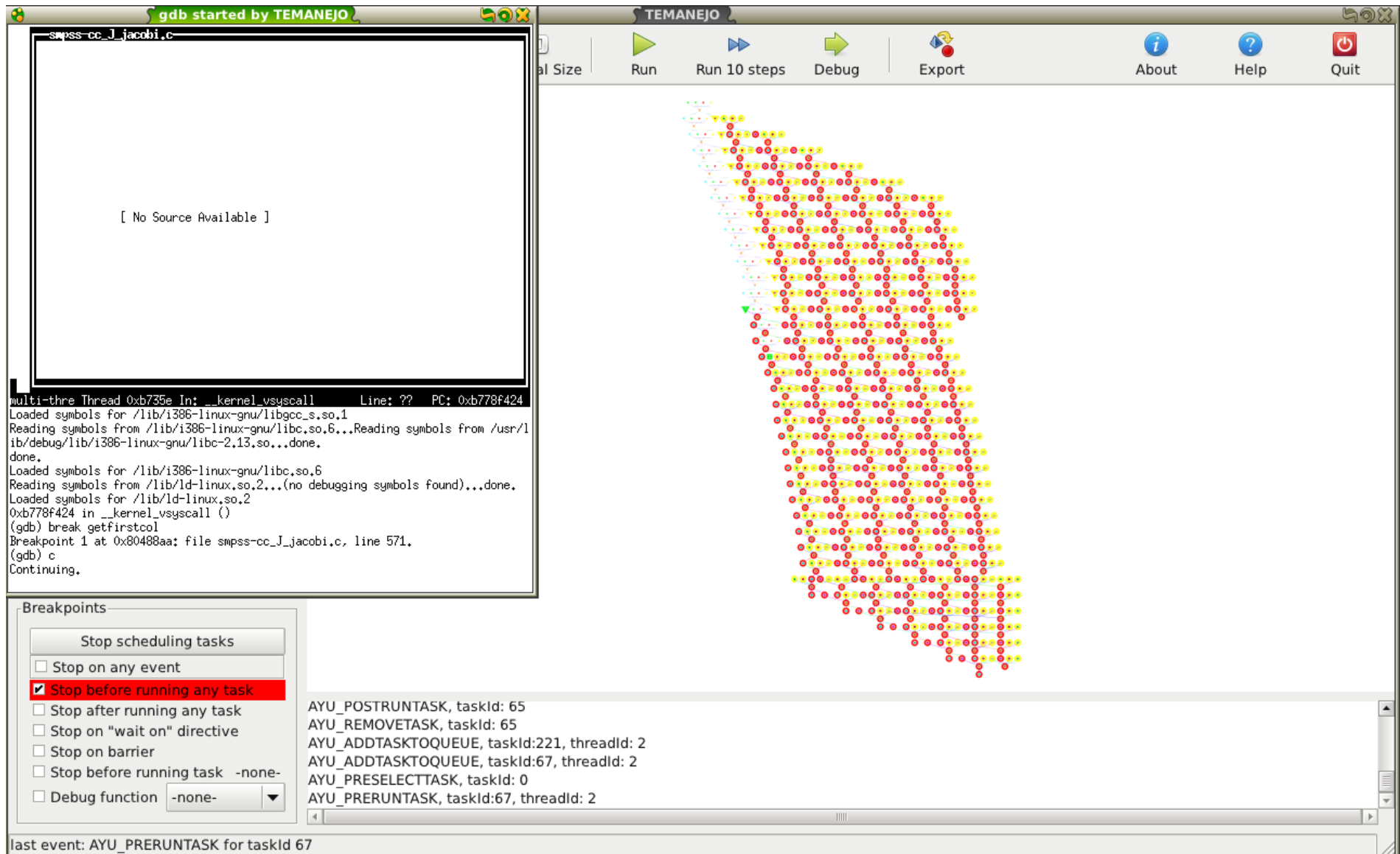
Below the table, there is a "Navigation" section with a "Goto task" input field (containing "1") and a "Clear marked nodes and edges" button. The "Breakpoints" section includes several checkboxes for stopping tasks at different points in the execution, with "Stop before running any task" currently selected. A "Stop scheduling tasks" button is also present.

At the bottom, a status bar indicates the last event: "last event: AYU_PRERUNTASK for taskId 76".

Press “Debug”



Set a breakpoint in gdb and continue



Press “Run” until breakpoint is reached, debug with gdb

The screenshot displays a GDB debugging session for a program named `smcss-cc_J_jacobi.c`. The left pane shows the source code with a breakpoint set at line 571. The right pane shows a memory dump of the matrix. The bottom pane shows the GDB console with commands and output, and a Breakpoints panel with 'Stop before running any task' selected.

Source Code (Left Pane):

```
560 void getfirstrow(float *A, float *v)
561 {
562     int j;
563     for (j = 0;
564         j < 32;
565         j++)
566         v[j] = A[j];
567 }
568 void getfirstcol(float *A, float *v)
569 {
570     int i;
571     for (i = 0;
572         i < 32;
573         i++)
574         v[i] = A[i * 32 + 32 - 1];
575 }
576 void jacobi(float *lefthalo, float *tophalo, float *righthalo, float *bottomhalo)
577 {
578     int i, j;
579     float tmp;
580     float left, top, right, bottom;
581     for (i = 0;
582         i < 32;
583         i++)
```

Memory Dump (Right Pane):

A memory dump showing a 32x32 matrix of floating-point values. The values are arranged in a pattern that suggests a sparse matrix, with many zeros and some non-zero values. The dump is organized into columns, with each column containing 32 values. The values are represented in hexadecimal and decimal notation.

GDB Console (Bottom Pane):

```
multi-thre Thread 0xb735e In: getfirstcol Line: 571 PC: 0x80488aa
done.
Loaded symbols for /lib/i386-linux-gnu/libc.so.6
Reading symbols from /lib/ld-linux.so.2...(no debugging symbols found)...done.
Loaded symbols for /lib/ld-linux.so.2
0xb778f424 in __kernel_vsyscall ()
(gdb) break getfirstcol
Breakpoint 1 at 0x80488aa: file smcss-cc_J_jacobi.c, line 571.
(gdb) c
Continuing.

Breakpoint 1, getfirstcol (A=0xa05c300, v=0xb6208000)
at smcss-cc_J_jacobi.c:571
(gdb)
```

Breakpoints Panel (Bottom Left):

- ☐ Stop scheduling tasks
- ☐ Stop on any event
- ☒ Stop before running any task
- ☐ Stop after running any task
- ☐ Stop on "wait on" directive
- ☐ Stop on barrier
- ☐ Stop before running task -none-
- ☐ Debug function -none-

Task List (Bottom Right):

```
AYU_PRESELECTTASK, taskId: 0
AYU_PRERUNTASK, taskId:176, threadId: 2
AYU_POSTRUNTASK, taskId: 176
AYU_REMOVETASK, taskId: 176
AYU_PRESELECTTASK, taskId: 0
AYU_PRERUNTASK, taskId:171, threadId: 2
```

Last Event (Bottom):

```
last event: AYU_PRERUNTASK for taskId 171
```

Uncheck “Stop on...” and Run until application finishes

The screenshot shows the TEMANEJO application interface. The top toolbar includes buttons for Connect, Preferences, Zoom In, Zoom Out, Best Fit, Normal Size, Run, Run 10 steps, Debug, Export, About, Help, and Quit.

On the left, a table summarizes function execution:

function	NQ	Q	R	F	total
clear	0	0	0	128	128
getfirstcol	0	0	0	992	992
getfirstrow	0	0	0	992	992
getlastcol	0	0	0	992	992
getlastrow	0	0	0	992	992
jacobi	0	0	0	1024	1024
total	0	0	0	5120	5120

Below the table is a legend for the state transition diagram:

- NQ: not queued (red square)
- Q: queued (yellow square)
- R: running (green square)
- F: finished (blue square)

The main area displays a large, complex state transition diagram with numerous nodes and edges, colored according to the legend.

On the bottom left, the Breakpoints section is visible:

- Stop scheduling tasks
- ☐ Stop on any event
- ☐ Stop before running any task
- ☒ Stop after running any task
- ☐ Stop on "wait on" directive
- ☐ Stop on barrier
- ☐ Stop before running task -none-
- ☐ Debug function -none-

The bottom right shows the last event: AYU_FINISH for taskId 0.

Todo's

- integration with MPI
- cleanup when starting new run
- connect and disconnect to AYUDAME
- connect to several hosts (nodes)
- set next task to run
- block task
- serialise (using `css_set_num_threads?`)