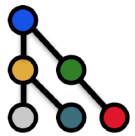
# HPCToolkit Graphical User Interface

Laksono Adhianto
Rice University

HPCToolkit Tutorial NERSC and OLCF
March 29, 2021
(Virtual)



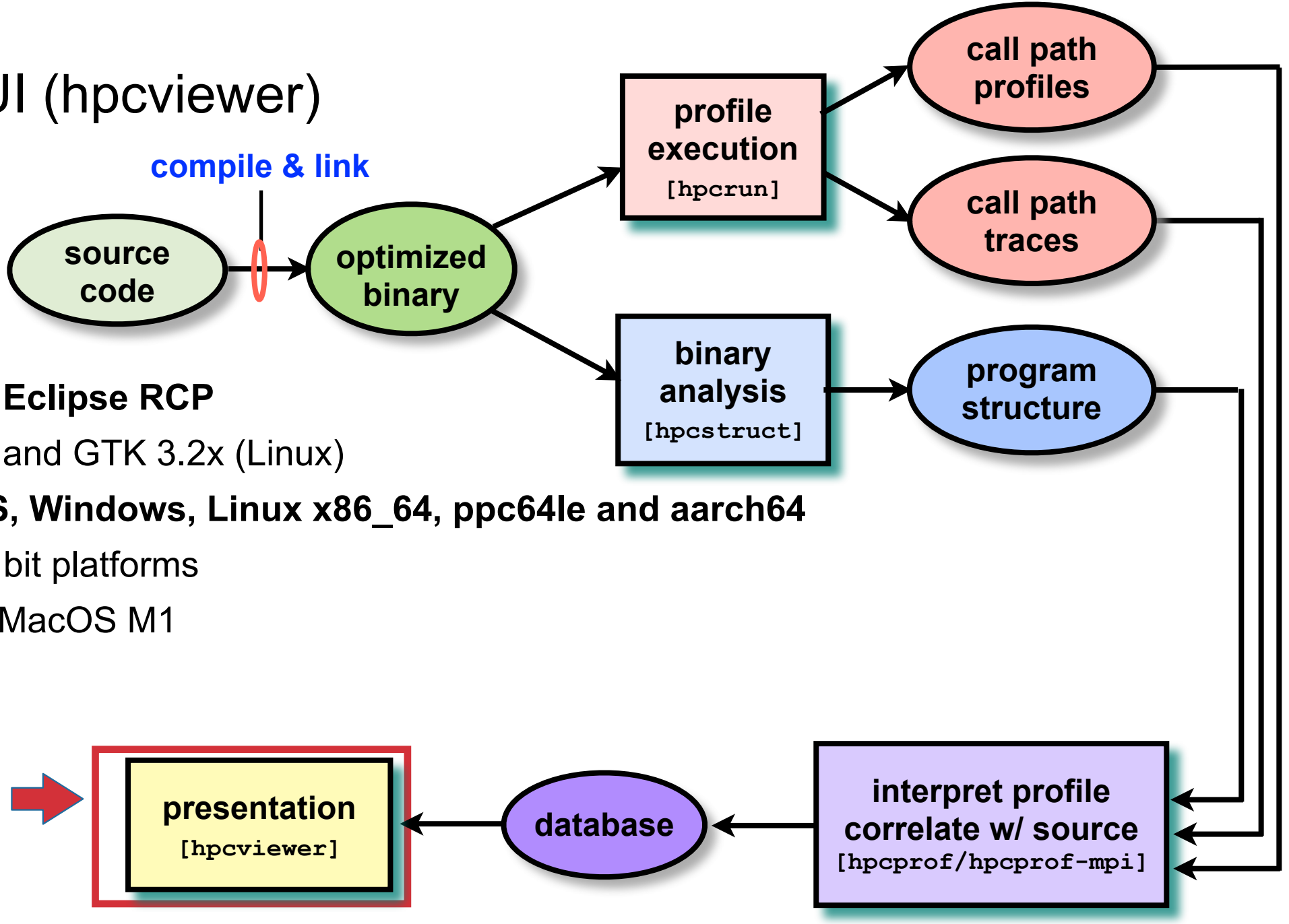




# Outline

- **Introduction to HPCToolkit GUI (hpcviewer)**
  - Overview of the HPCToolkit GUI
  - Installing and launching hpcviewer
  - HPCToolkit database
- **Working with hpcviewer**
  - Introduction to the Profile view
  - Introduction to the Trace view
  - Tips using hpcviewer
- **Demo**
  - On remote machines (e.g cori and summit)
  - On local machine (laptop)

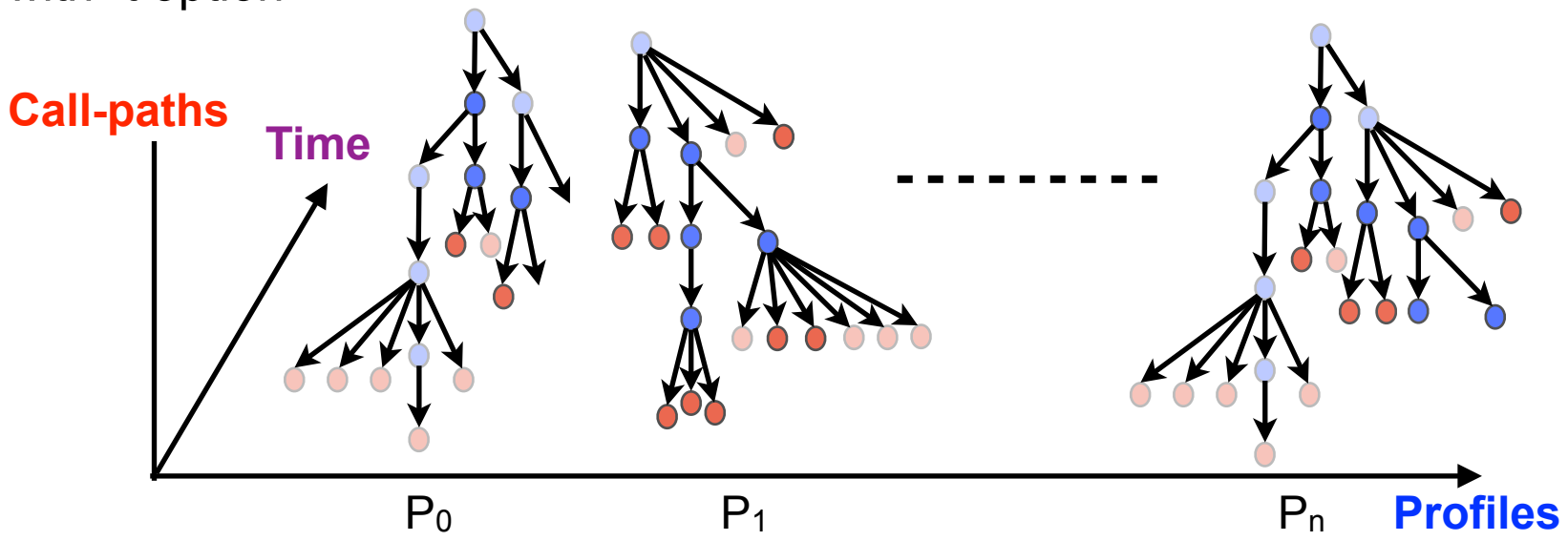
# HPCToolkit GUI (hpcviewer)




- **Based on Java and Eclipse RCP**
  - Requires Java 11 and GTK 3.2x (Linux)
- **Available on MacOS, Windows, Linux x86_64, ppc64le and aarch64**
  - No support for 32 bit platforms
  - Not tested yet on MacOS M1

# HPCToolkit Database

- **HPCToolkit database contains four information:**
  - **Call-paths**: the union of all functions/loops/statements as each measurement taken
  - **Profiles**: a list of threads, processes and/or GPU streams
  - **Metrics**: a set of hpcrun events (-e option)
    - Exclusive (E): the quantity of the metric measured for a scope alone
    - Inclusive (I): the value measured for that scope as well as costs incurred by any functions it calls.
  - **Time**: a sequence of time of the sample
    - Available when run `hpcrun` with -t option



# Installing hpcviewer

- Already available on Cori (NERSC) and Summit (ORNL)
  - Type: **`module load hpcviewer/2021.03.01`**
- To install locally:
  - Download prebuilt binaries http://hpctoolkit.org/download.html
    - Linux and Windows: download directly from the web browser
    - MacOS: download via **curl** program to bypass Apple Gatekeeper
  - Build with command line
    - Requires Java 11 and Apache Maven 3.6 or newer
    - See the instructions at http://github.com/hpctoolkit/hpcviewer.e4
  - Linux only: **`spack install hpcviewer`**
  - **Caveat**: need to copy the database to the machine where hpcviewer is installed

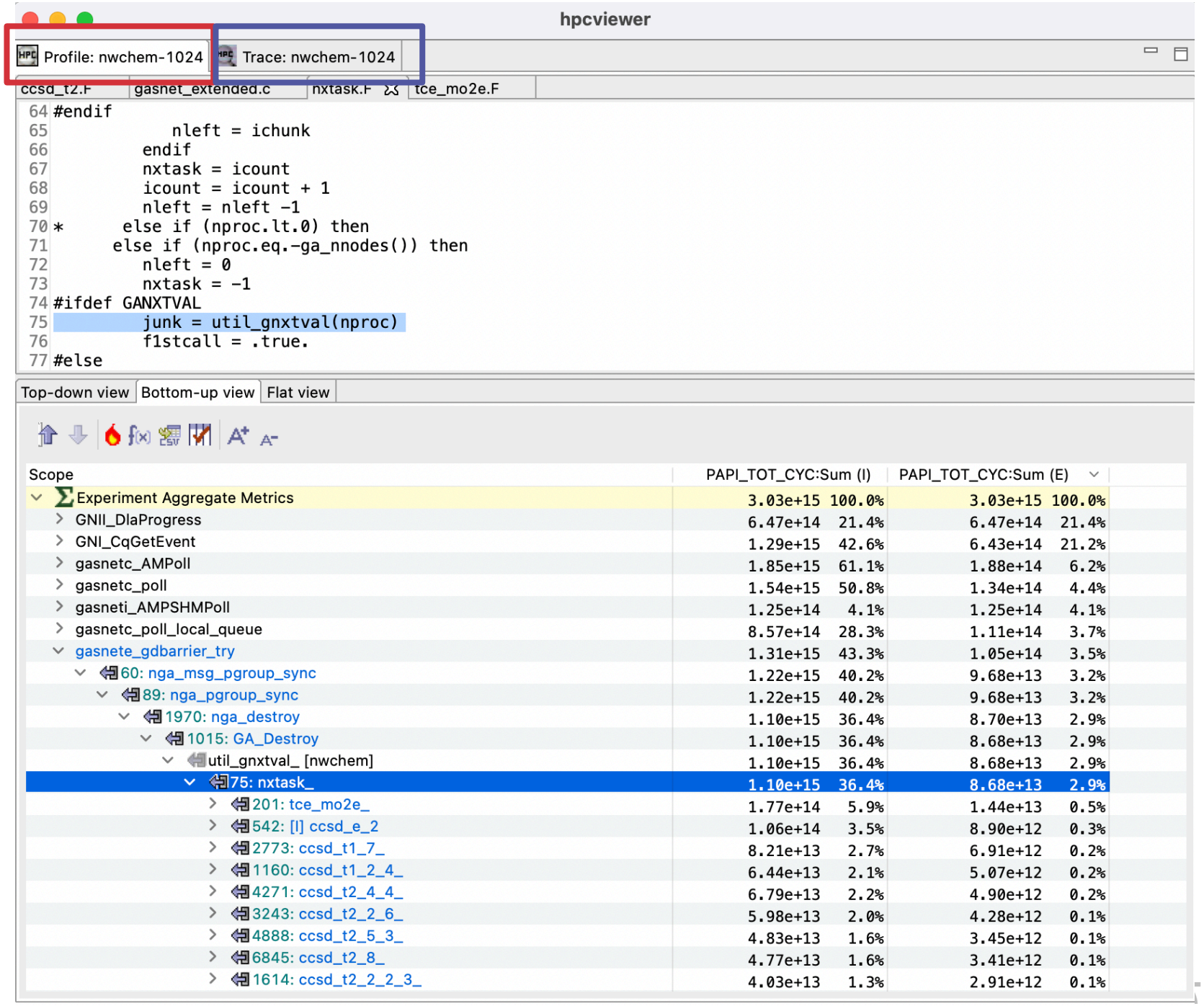
# Launching hpcviewer

- On Linux:
  - Type: **`hpcviewer [options] [database]`**
- On Windows and MacOS:
  - Simply click the hpcviewer icon
  - Windows command line: **`hpcviewer.exe`**
  - MacOS command line: **`open hpcviewer.app`**

# Modes in hpcviewer

**Two modes:**

- **Profile**: presents the summary of application performance with different perspectives: *top-down*, *bottom-up* and *flat*
- **Trace**: presents program traces in a top-down fashion. This view is only visible if the trace information is available in the database.
  Previously the trace view was a separate program



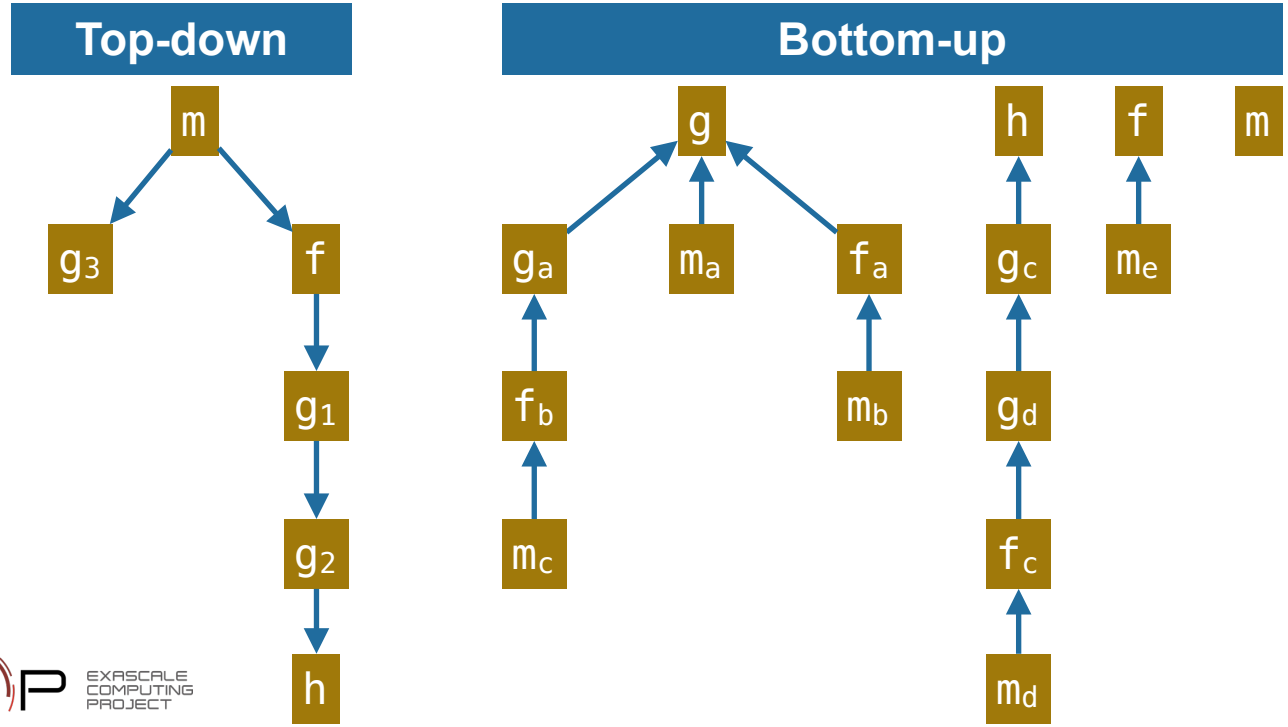
# Profile View

- **Top-down view:** presents dynamic calling contexts (call paths) in which costs were incurred
- **Bottom-up view:** presents costs by looking upward along call paths
- **Flat view:** presents costs based on the structure of an application

```
file1.c
// m: main routine
m () {
        f ();
        g ();
}
f () {
        g ();
}
```

```
file2.c
g () {
        if ( . . ) g ();
        if ( . . ) h ();
}

h () {
}
```

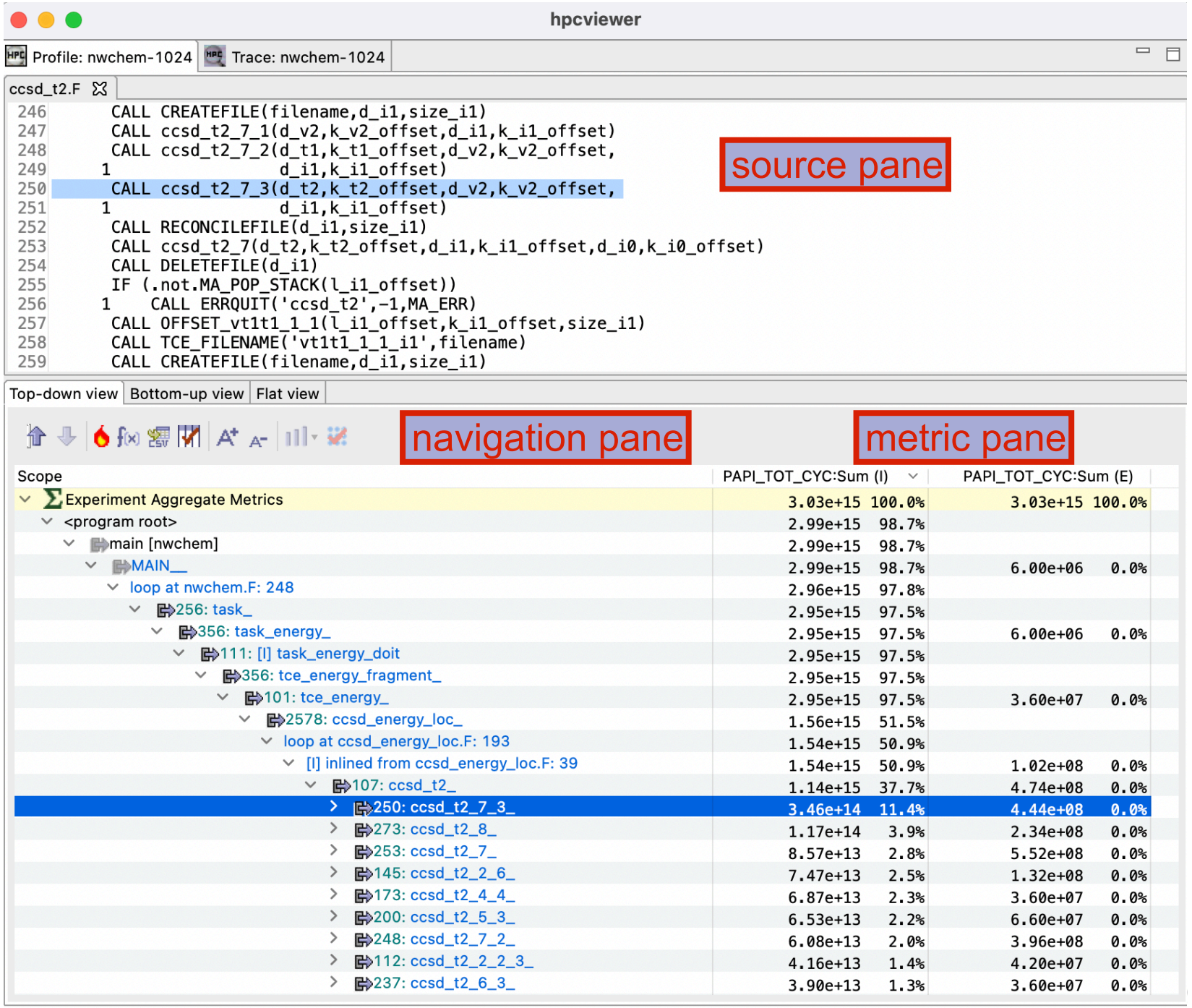


# Top-down view

## Tips

- Click the **hot-path** button to automatically drill down the tree. It is an easy way to find performance bottlenecks for the selected metric
- Clicking the call-site icon or the line number will highlight the call location on the source pane

- **function calls in full context**
- **inlined procedures**
- **inlined macros/templates**
- **sequential loops**
- **outlined OpenMP loops (not shown)**
- **Line statements (not shown)**




```
246       CALL CREATEFILE(filename,d_i1,size_i1)
247       CALL ccsd_t2_7_1(d_v2,k_v2_offset,d_i1,k_i1_offset)
248       CALL ccsd_t2_7_2(d_t1,k_t1_offset,d_v2,k_v2_offset,
249      1                 d_i1,k_i1_offset)
250       CALL ccsd_t2_7_3(d_t2,k_t2_offset,d_v2,k_v2_offset,
251      1                 d_i1,k_i1_offset)
252       CALL RECONCILEFILE(d_i1,size_i1)
253       CALL ccsd_t2_7(d_t2,k_t2_offset,d_i1,k_i1_offset,d_i0,k_i0_offset)
254       CALL DELETEFILE(d_i1)
255       IF (.not.MA_POP_STACK(l_i1_offset))
256      1    CALL ERRQUIT('ccsd_t2',-1,MA_ERR)
257       CALL OFFSET_vt1t1_1_1(l_i1_offset,k_i1_offset,size_i1)
258       CALL TCE_FILENAME('vt1t1_1_1_i1',filename)
259       CALL CREATEFILE(filename,d_i1,size_i1)
```

| Scope | PAPI_TOT_CYC:Sum (I) | | PAPI_TOT_CYC:Sum (E) | |
|---|---|---|---|---|
| Experiment Aggregate Metrics | 3.03e+15 | 100.0% | 3.03e+15 | 100.0% |
| <program root> | 2.99e+15 | 98.7% | | |
| main [nwchem] | 2.99e+15 | 98.7% | | |
| MAIN__ | 2.99e+15 | 98.7% | 6.00e+06 | 0.0% |
| loop at nwchem.F: 248 | 2.96e+15 | 97.8% | | |
| 256: task_ | 2.95e+15 | 97.5% | | |
| 356: task_energy_ | 2.95e+15 | 97.5% | 6.00e+06 | 0.0% |
| 111: [I] task_energy_doit | 2.95e+15 | 97.5% | | |
| 356: tce_energy_fragment_ | 2.95e+15 | 97.5% | | |
| 101: tce_energy_ | 2.95e+15 | 97.5% | 3.60e+07 | 0.0% |
| 2578: ccsd_energy_loc_ | 1.56e+15 | 51.5% | | |
| loop at ccsd_energy_loc.F: 193 | 1.54e+15 | 50.9% | | |
| [I] inlined from ccsd_energy_loc.F: 39 | 1.54e+15 | 50.9% | 1.02e+08 | 0.0% |
| 107: ccsd_t2_ | 1.14e+15 | 37.7% | 4.74e+08 | 0.0% |
| 250: ccsd_t2_7_3_ | 3.46e+14 | 11.4% | 4.44e+08 | 0.0% |
| 273: ccsd_t2_8_ | 1.17e+14 | 3.9% | 2.34e+08 | 0.0% |
| 253: ccsd_t2_7_ | 8.57e+13 | 2.8% | 5.52e+08 | 0.0% |
| 145: ccsd_t2_2_6_ | 7.47e+13 | 2.5% | 1.32e+08 | 0.0% |
| 173: ccsd_t2_4_4_ | 6.87e+13 | 2.3% | 3.60e+07 | 0.0% |
| 200: ccsd_t2_5_3_ | 6.53e+13 | 2.2% | 6.60e+07 | 0.0% |
| 248: ccsd_t2_7_2_ | 6.08e+13 | 2.0% | 3.96e+08 | 0.0% |
| 112: ccsd_t2_2_2_3_ | 4.16e+13 | 1.4% | 4.20e+07 | 0.0% |
| 237: ccsd_t2_6_3_ | 3.90e+13 | 1.3% | 3.60e+07 | 0.0% |

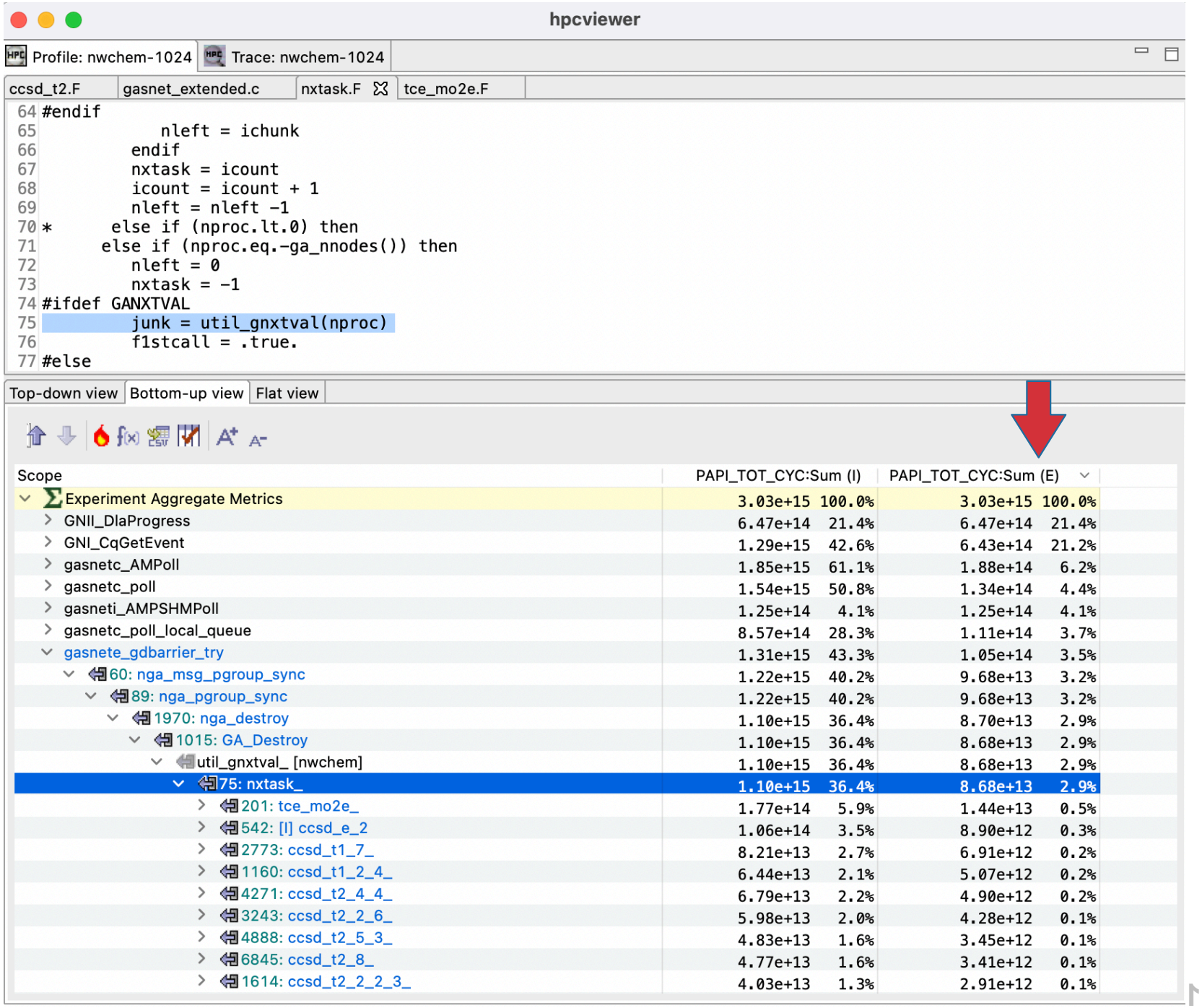
# Bottom-up view

## Tips

- Click the header of a metric column to sort the column
- Sorting by an exclusive metric is very useful to find the costliest functions
- Click the **hot-path** 🔥 button to see the most important way or ways the selected function was reached

## Caveats

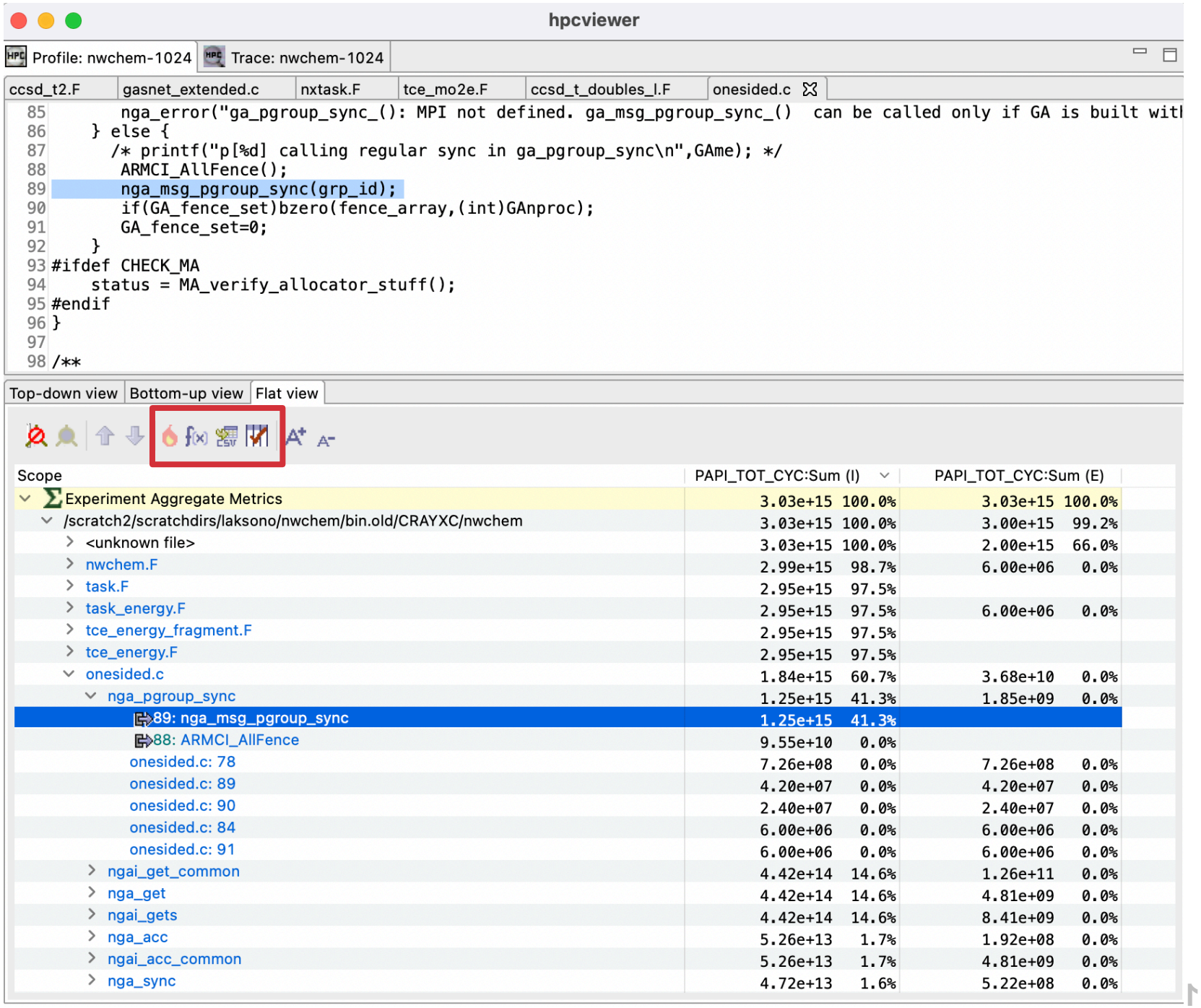
- Suffix “Sum” on the column header means the value is the sum over all ranks/threads, <u>including helper threads</u>



# Flat view

## Tips

- Use Flat view to identify overheads of the libraries (communication, I/O, OpenMP libraries, …)
- If there are too many metrics, you can hide some metric columns by clicking the button
- Click the f(x) button to create a derived metric from existing metrics



# Creating a User-Defined Metric

- Assume the database has 2 metrics:
  - PAPI_TOT_CYC has the metric-id **2048**
  - PAPI_FP_INS has the metric-id **2050**
- To compute the inclusive metric of “Cycle Per Instruction (CPI)” :**`$2048 / $2050`**
- Two ways to reference a metric:
  - Using the **$** : a point-wise value of a metric at a node in the tree
  - Using **@** : the aggregate metric value at the root of the tree

Top-down view | Bottom-up view | Flat view

| Scope | CPI (I) | PAPI_TOT_CYC:Sum (I) | | PAPI_FP_INS (proxy):Sum (I) | |
|---|---|---|---|---|---|
| ccsd_t.F | 3.42 | 8.53e+14 | 47.6% | 2.49e+14 | 97.0% |
| ccsd_t_doubles_l.F | 3.34 | 8.25e+14 | 46.0% | 2.47e+14 | 96.1% |
| ccsd_t_doubles_l_2_ | 4.02 | 6.73e+14 | 37.5% | 1.67e+14 | 65.2% |
| ccsd_t_doubles_l_ | 1.92 | 1.52e+14 | 8.5% | 7.94e+13 | 30.9% |
| ccsd_t_doubles_l_1_ | 1.92 | 1.52e+14 | 8.5% | 7.94e+13 | 30.9% |
| loop at ccsd_t_doubles_l.F: 235 | 1.92 | 1.52e+14 | 8.5% | 7.94e+13 | 30.9% |
| loop at ccsd_t_doubles_l.F: 235 | 1.92 | 1.52e+14 | 8.5% | 7.94e+13 | 30.9% |
| 482: [I] sd_t_d1_9 | 1.99 | 1.96e+13 | 1.1% | 9.83e+12 | 3.8% |
| 322: [I] sd_t_d1_1 | 2.01 | 1.93e+13 | 1.1% | 9.61e+12 | 3.7% |
| 422: [I] sd_t_d1_6 | 1.94 | 1.89e+13 | 1.1% | 9.75e+12 | 3.8% |
| loop at ccsd_t_doubles_l.F: 322 | 1.91 | 1.88e+13 | 1.0% | 9.82e+12 | 3.8% |
| loop at ccsd_t_doubles_l.F: 322 | 1.91 | 1.88e+13 | 1.0% | 9.82e+12 | 3.8% |
| loop at ccsd_t_doubles_l.F: 322 | 1.91 | 1.88e+13 | 1.0% | 9.82e+12 | 3.8% |
| loop at ccsd_t_doubles_l.F: 322 | 1.91 | 1.88e+13 | 1.0% | 9.81e+12 | 3.8% |

**Creating a derived metric**

A derived metric is a spreadsheet-like formula using other metrics (variables), operators, functions, and numerical constants.

Derived metric definition

Name: CPI (I)

Formula: $2048/$2050

There are two kinds of metric variables: point-wise and aggregate. The former is like a spreadsheet cell, the latter is like a spreadsheet-column sum. To form a variable, prepend '$' and '@', respectively, to a metric id. For instance, the formula

(($2 - $1) * 100.0) / @1

divides the scaled difference of the point-wise metrics 2 and 1 by the aggregate value of metric 1.

Assistance:

Metrics: 2050: PAPI_FP_INS (proxy):Sum (I) | Point-wise | Aggregate

Functions: stdev(x1, x2, ..., xn) | Insert function

Operators: ( ) + - * / ^

Advanced options

- [ ] Augment metric value display with a percentage relative to column total
- ( ) Default format
- ( ) Display metric value as percent
- (•) Custom format %6.2f

The format is based on java.util.Formatter class which is almost equivalent to C's printf format.
Example: '%6.2f ' will display 6 digit floating-points with 2 digit precision.

Cancel | OK

# Metric description

## Tips

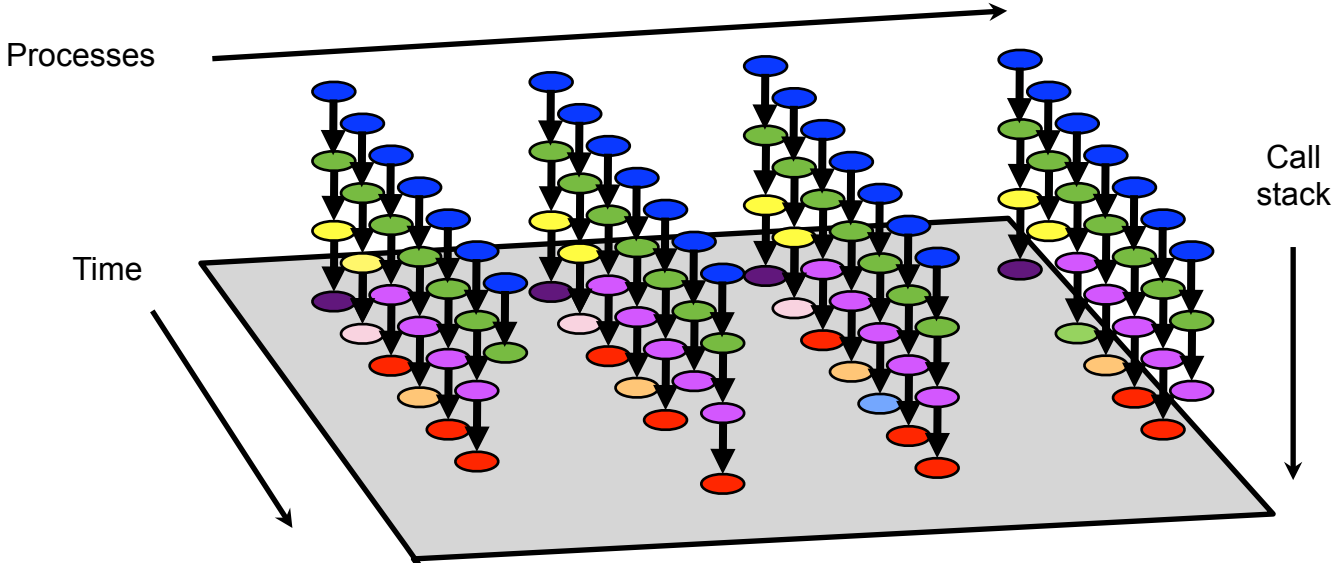
- Click “**View - Show metrics**” menu to find the metric descriptions
- The **Metric property** window also allows to edit the metric’s label and the formula of the user-defined metric
- Hovering the mouse over a metric-column header will display a tooltip of the metric’s description

## Caveats

- User-defined metrics are not persistent. You need to create them again every time you open databases

# Trace View: Understanding Temporal Behavior

- **Profiling compresses out the temporal dimension**
  - Temporal patterns, e.g. serial sections and dynamic load imbalance are invisible in profiles
  - We need to explore temporal behavior of the application
- **What can we do? Trace call path samples**
  - N times per second, take a call path sample of each thread
    - Use hpcrun trace option: `hpcrun -t …`
  - View how the execution evolves left to right
  - hpcviewer assigns each procedure a color; view a depth slice of an execution

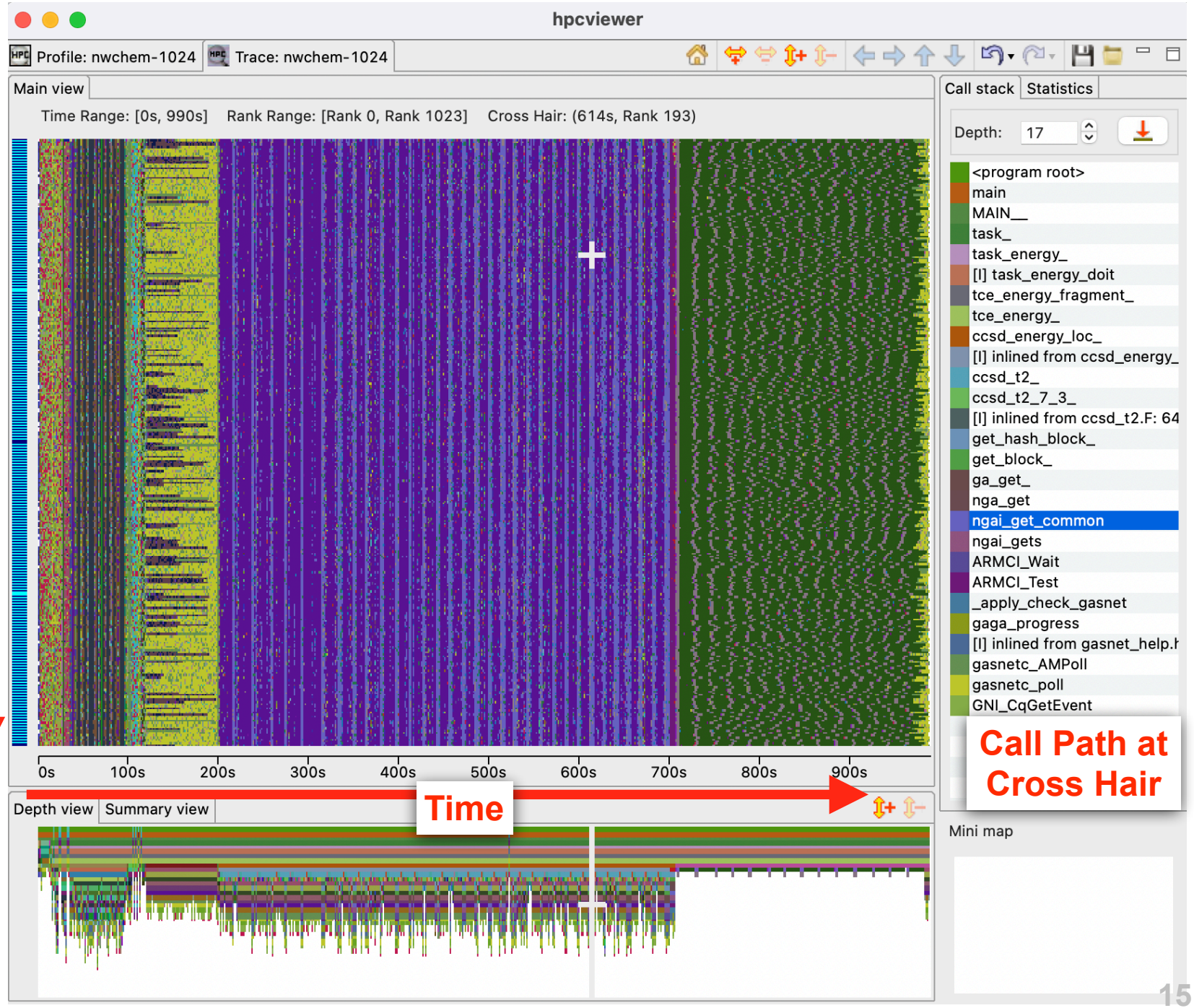


# Trace View

## Tips

- You can zoom horizontally by selecting a region in the Depth view
- Click the **Max-depth** button to set to the maximum call-stack depth
- Use the **Undo** button to return to the previous region

**Depth view**: the call stack across the current displayed time range in a specified rank



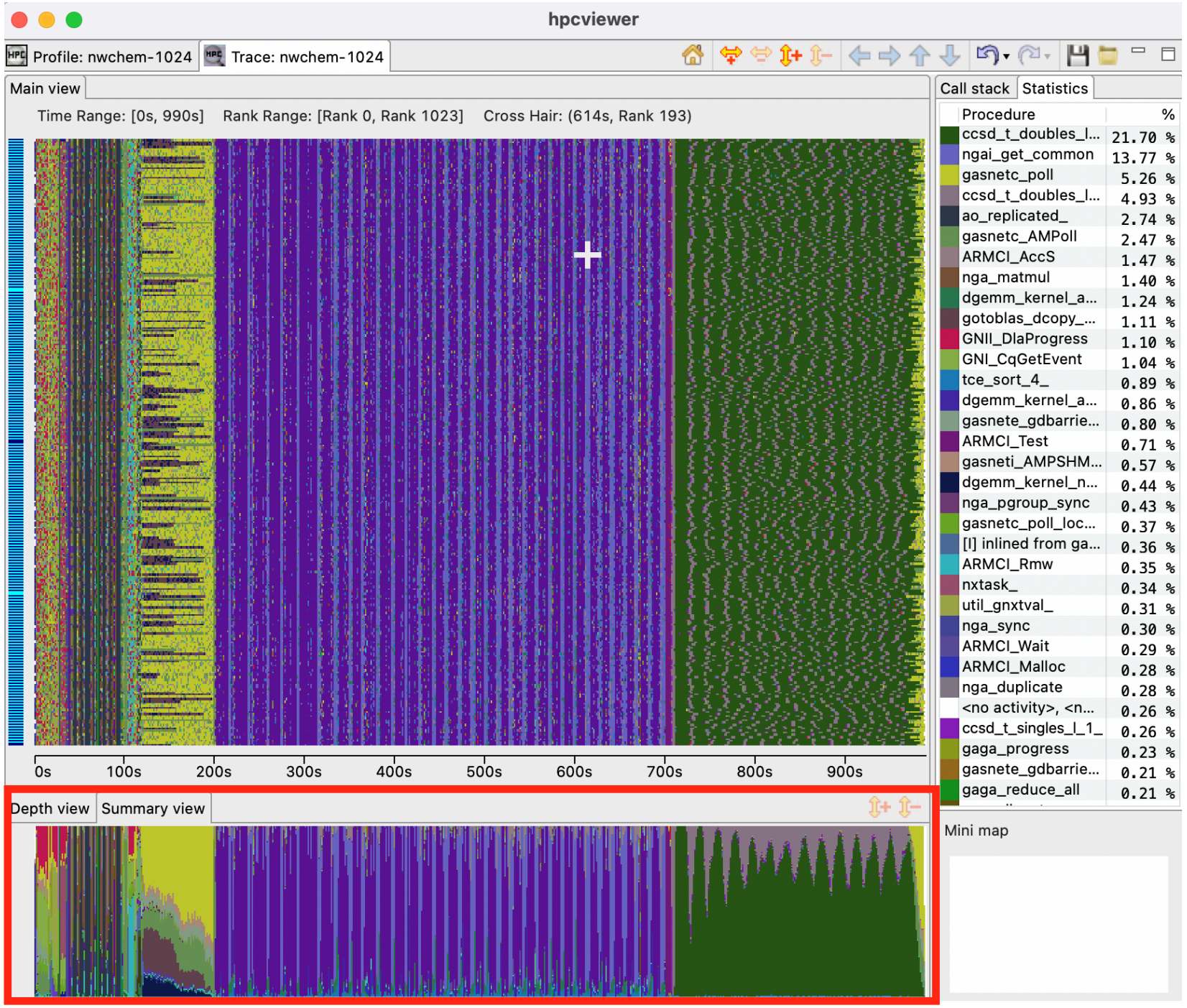
# Trace View

## Tips

- Summary view: can be used to identify load imbalance
- You can zoom horizontally by selecting a region in the Summary view
- Statistics view: clicking the column header will sort based on the column

**Summary view**: the projection of number of calls of across the current displayed time-range and rank

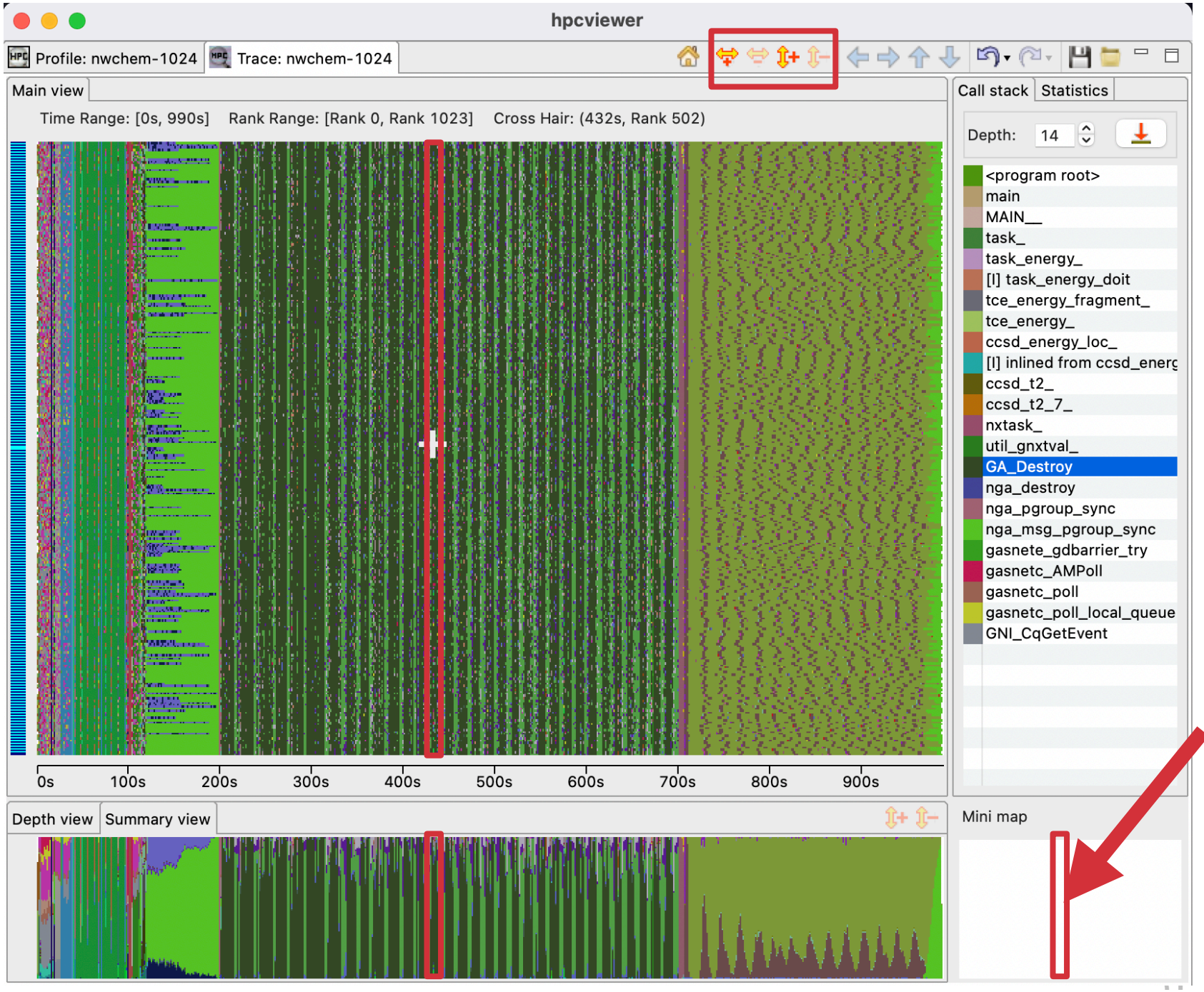
**Statistics view**: the proportion of number of samples across the current displayed time-range and rank



# Trace View: Zoom

**Different ways to zoom**

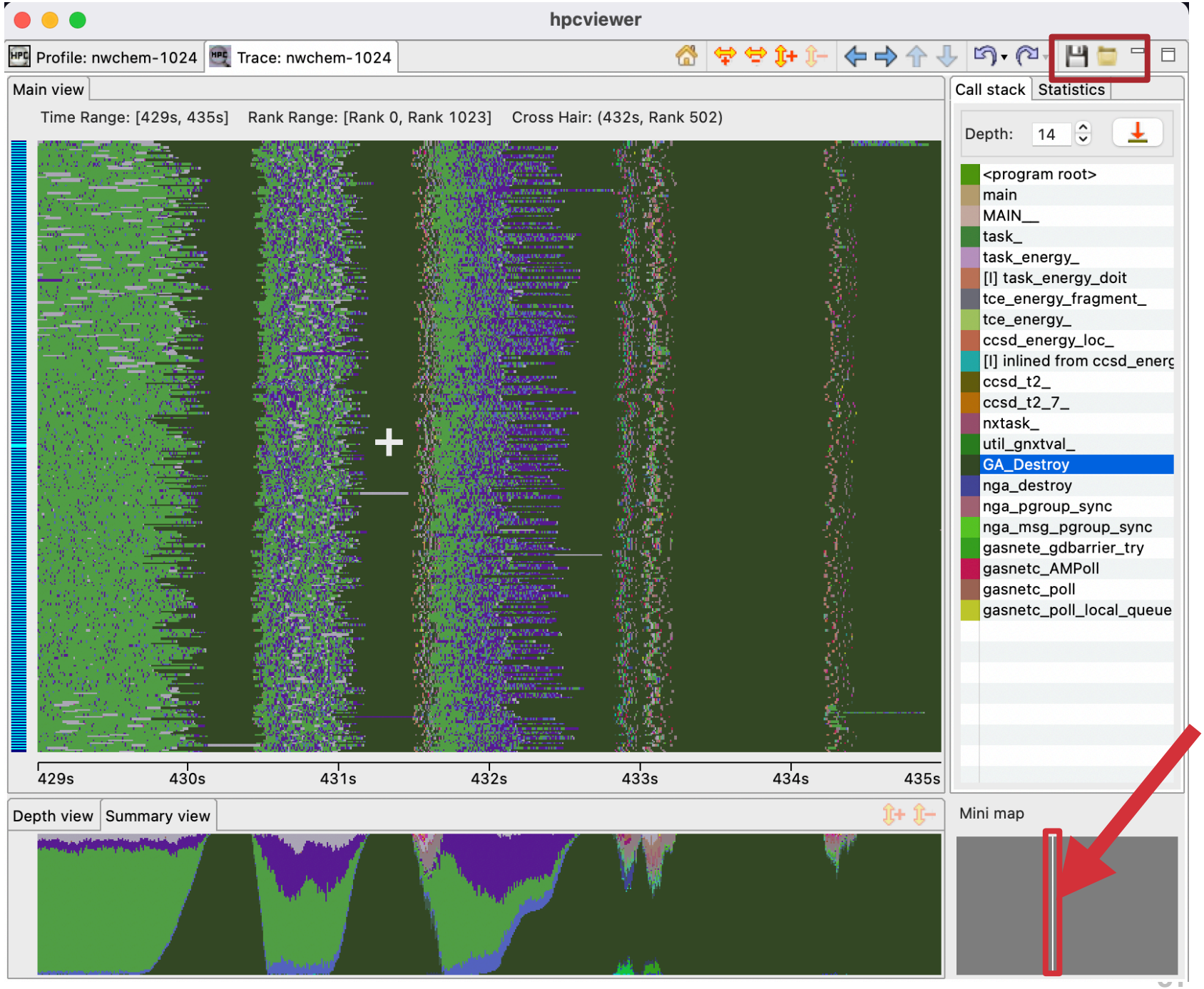
- Click the zoom buttons on the top-left toolbar
- Select a region in the Main view
- Select a time range in the Depth or Summary View
- Select a region in the Mini map (only if the view is already zoom-in)



# Trace View: Zoom

## Tips

- To save the current region, click the **Save** button at the top-left toolbar.
- To load the previous saved region, click the **Load** button, and select the file.



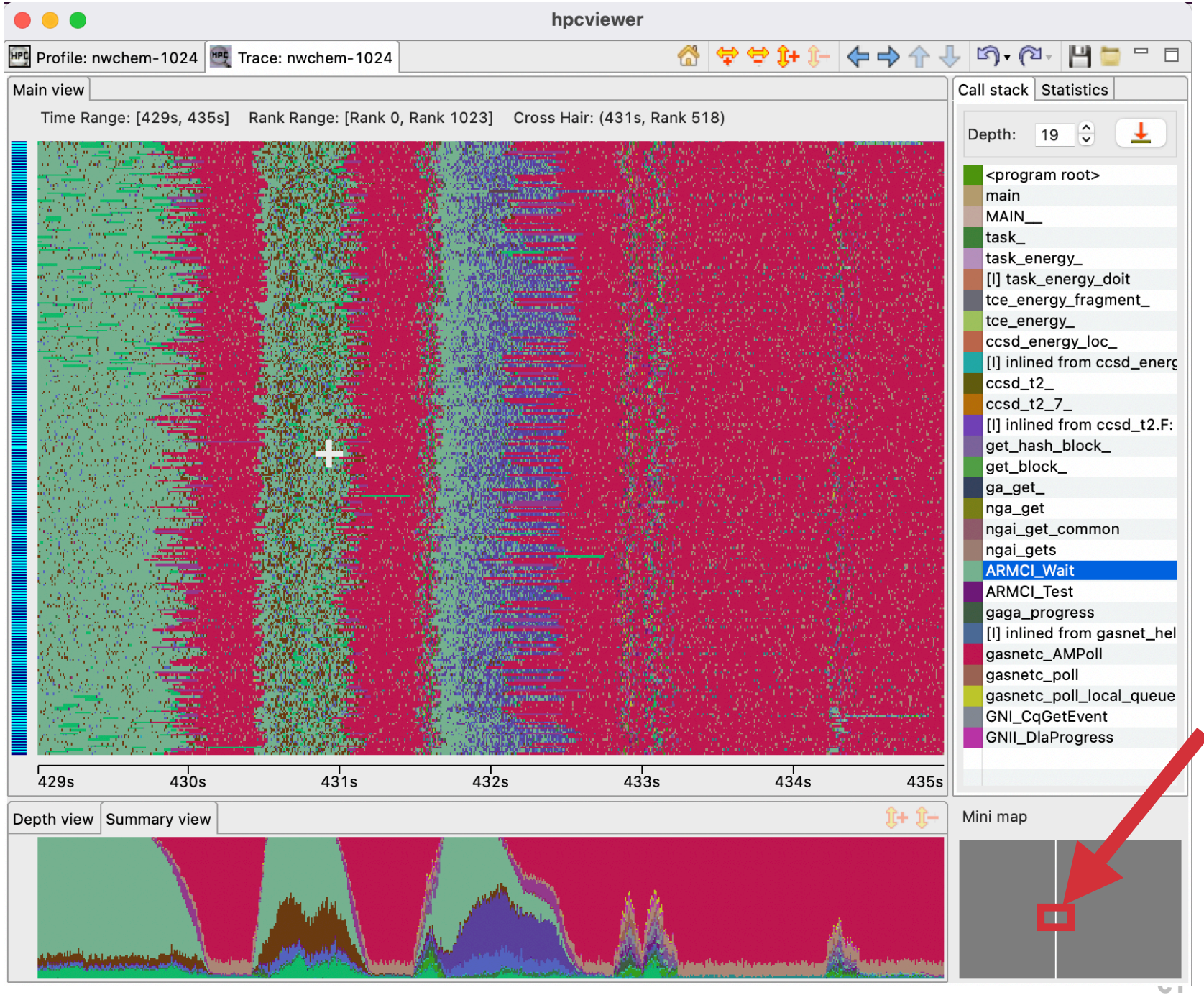
# Trace View: Zoom

## Caveats

- Colors are generated randomly
- Procedure's color can be different every time the database is opened

## Tips

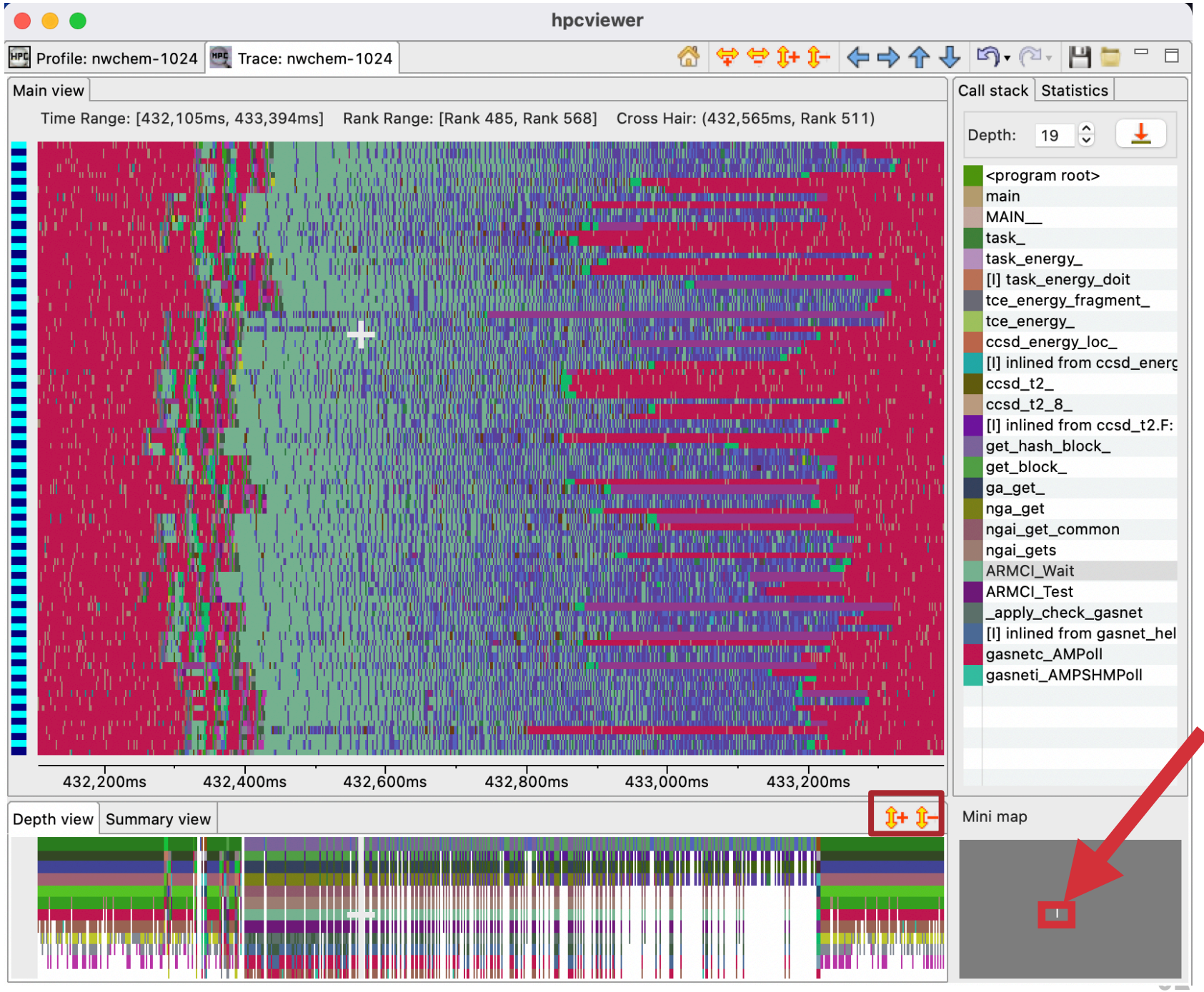
- You can assign procedures to a certain color by selecting "View - Procedure-Color Map" menu
- It can be useful to assign a color to all routines in the OpenMP runtime, e.g. matching ∗**kmp**∗ to assess how much time is spent in the runtime
- The user-defined color mapping is **persistent** across different hpcviewer instances



# Trace View: Zoom

## Tips

- You can zoom the Depth view by clicking the “Zoom” buttons at the bottom right toolbar
- Unusual changes or clustering of deep call stacks can indicate behaviors of potential interest



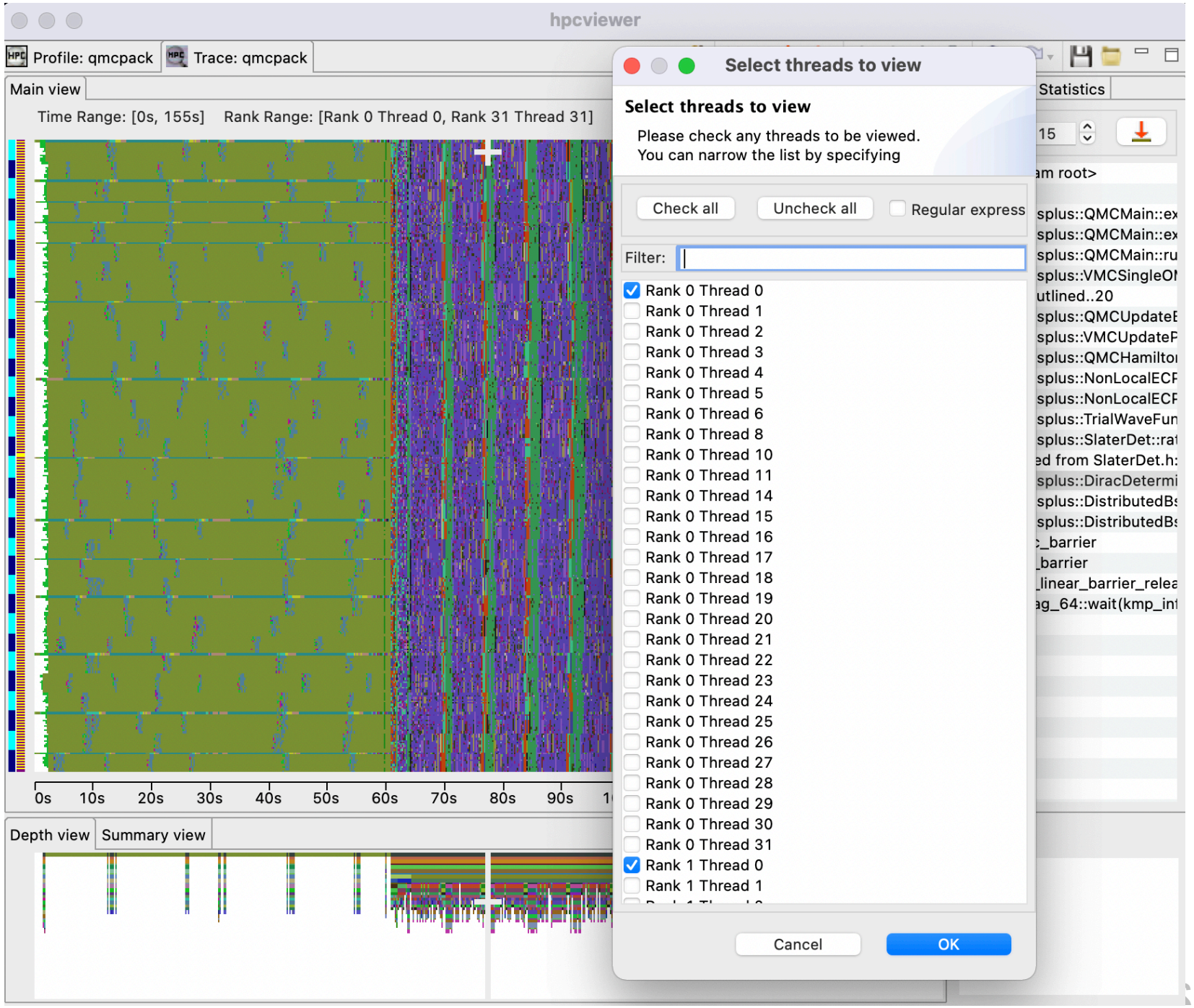
# Trace View: Filter

## Hiding processes/threads

- Useful to view only certain processes or threads
- Select “Filter - Filter ranks” menu
- Check the threads of interest

**Example #1**: to view only the main threads:

- Click “Uncheck all” button
- Type “Thread 0” in the Filter field
- Click “Check all” button
- Click “Ok” button



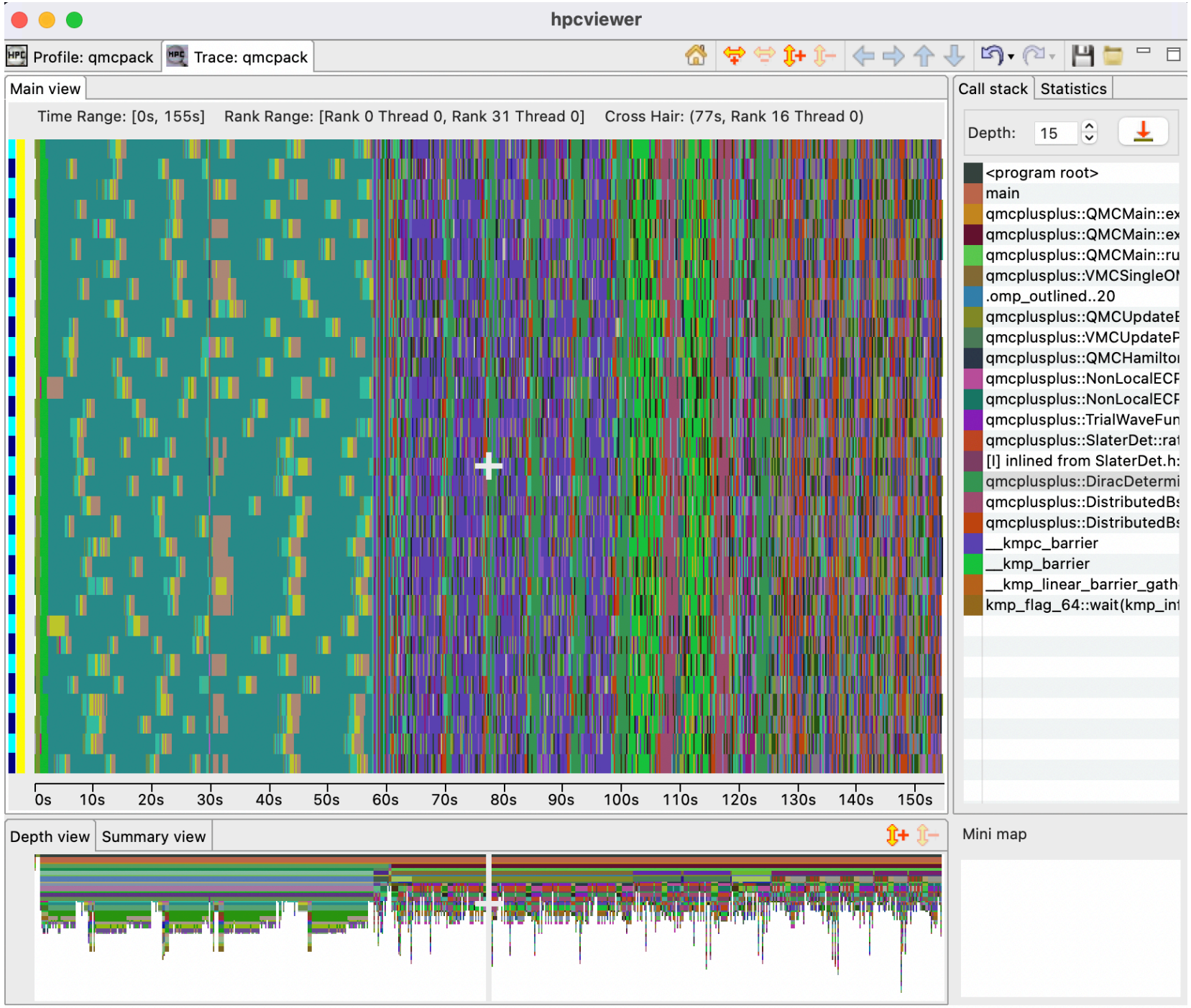
# Trace View: Filter

**Displaying only the main threads**

- Useful to see the interaction between MPI processes

**Example #2**: to view only the helper threads:

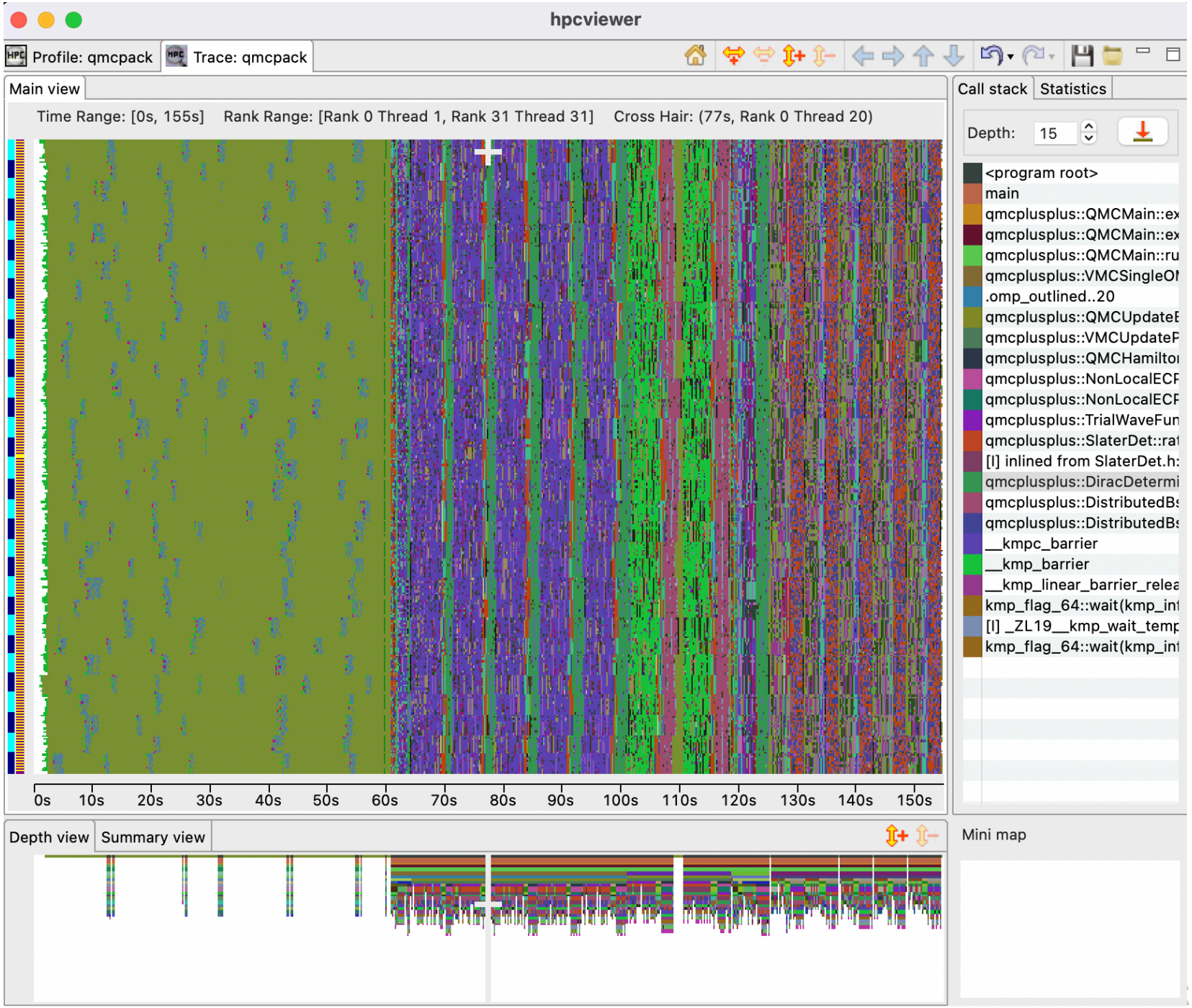
- Click “Check all” button
- Type “Thread 0” in the Filter field
- Click “Uncheck all” button
- Click “Ok” button



# Trace View: Filter

## Tips

- Rank filter is useful to hide helper threads. OpenMPI has 2 helper-threads for each MPI process



# Demo

# Troubleshooting

# Changing the maximum size of memory allocation pool

- On Linux: hpcviewer [options] [database]

  -h --help Print a help message.

  -jh, --java-heap <size>
- On Windows:
  - Change the value of -Xmx2G in hpcviewer/hpcviewer.ini file
- On MacOS:
  - Change the value of -Xmx2G in hpcviewer.app/Contents/Eclipse/hpcviewer.ini file

# Coping with Temporary Idiosyncrasies in HPCToolkit

- **I brought up hpcviewer and I am staring at a blank pane**
  - click in it! sometimes Eclipse doesn't refresh the pane when it should