Lab 3: Analysis of parallel strategies:

the computation of the Mandelbrot set

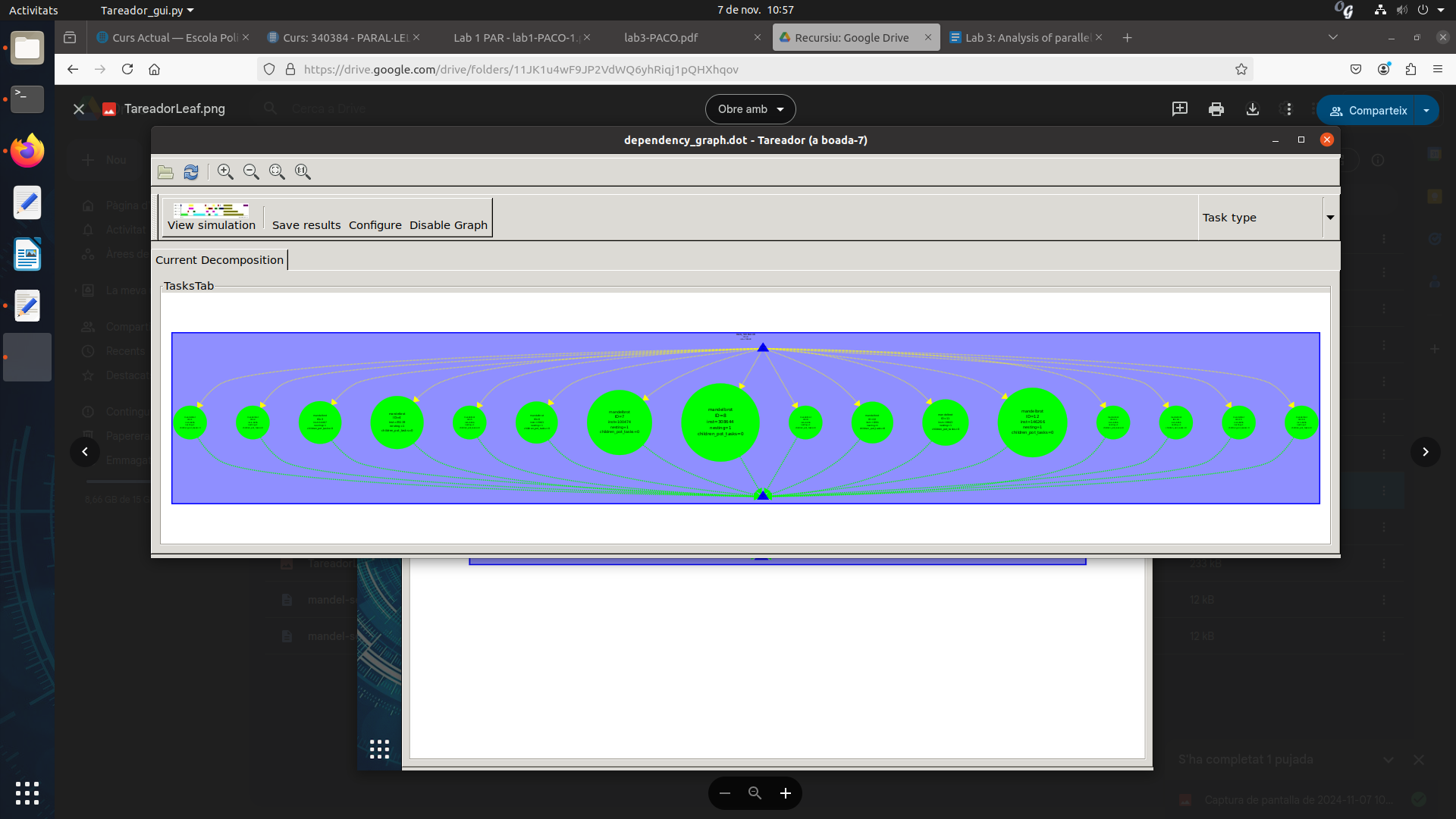
| **Task decomposition** | **Strategy** | **Run arguments** | **T1(ns)** | **Tinf(ns** | **Parallelism** | **Load Unbalance (Yes/no) Why?** |
| --- | --- | --- | --- | --- | --- | --- |
| Interative | Original |  | 705,540,001 | 308,750,001 | 2,285 | Yes |
| Original | -d | 729,092,001 | 728,904,001 | 1 | Yes |
| Original | -h | 713,860,001 | 665,997,001 | 1,072 | Yes |
| Finer grain |  | 705,557,001 | 221,147,001 | 3,19 | Yes |
| Column |  | 589,639,001 | 428,685,001 | 1,375 | Yes |
| Recursive | Leaf |  | 450,920,001 | 922,779,001 | 2,05 | Yes |
| Tree |  | 922,779,001 | 559,979,001 | 1,647 | Yes |

Load unbalanced: Tots els grafs estan desbalancejats ja que no totes les tasques tenen la mateixa càrrega de treball, hi ha tasques que en fan més, i per tant en el graf se simbolitzen amb una figura geometria més gran, i tasques que en fan de menys, i per tant la seva geometria al graf és més petita.

**ITERATIU:**

Primer hem executat el codi que ens proporcionen sense fer cap canvi.

./run-tareador.sh mandel-seq-iter-tar



Ara hem de executar

./run-tareador.sh mandel-seq-iter-tar -d

per mostrar l’imatge de Mandelbrot.

En aquest cas si que hem fet canvis al codi.

Hem afegit cada cop que s’utiltzaba la variable display.

Per exemple:

tareador\_disable\_object(&X11\_COLOR\_fake);

/\* Scale color and display point \*/

if (setup\_return == EXIT\_SUCCESS) {

XSetForeground(display, gc, color);

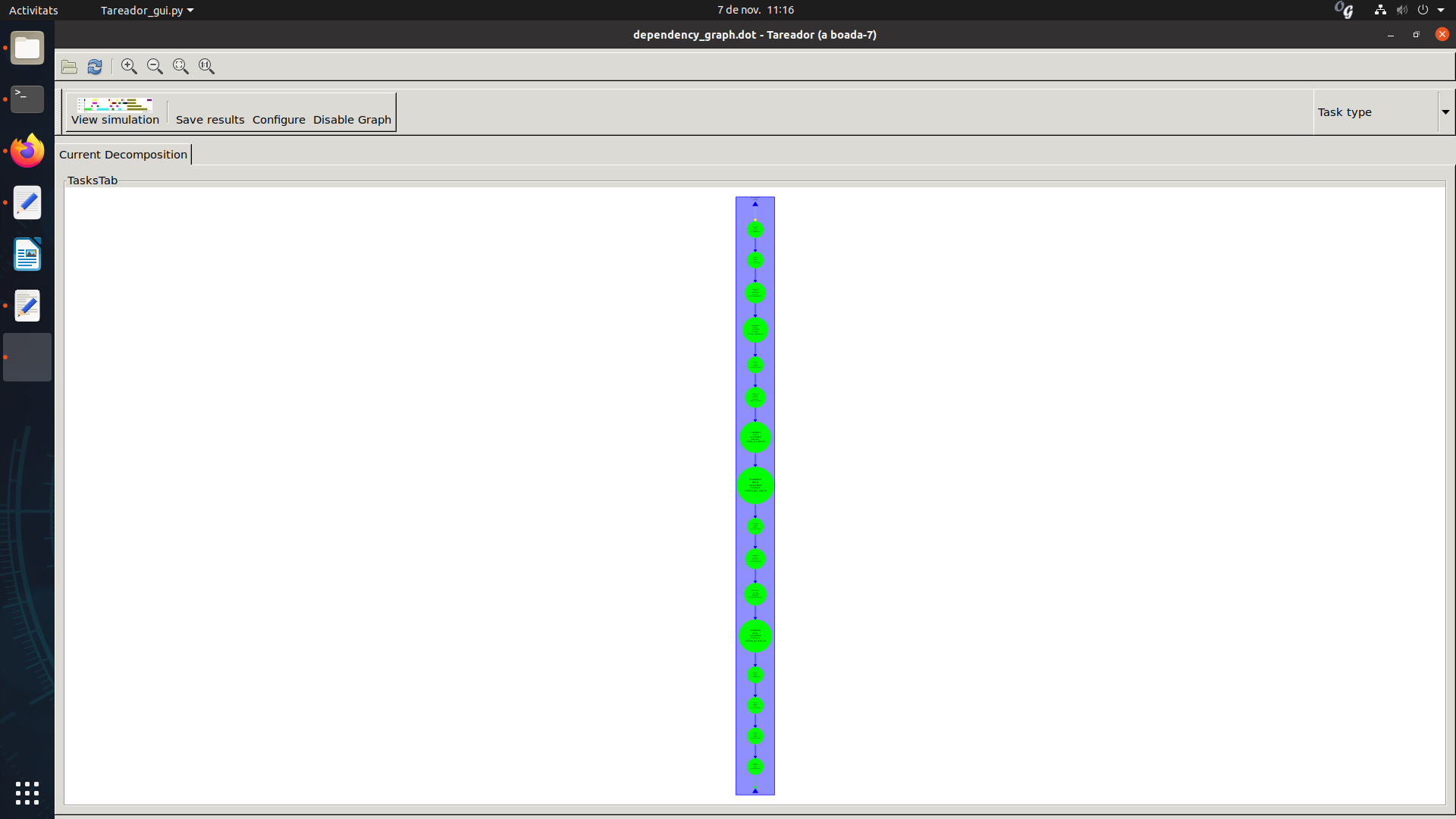
XDrawPoint(display, win, gc, px, py);

}

tareador\_enable\_object(&X11\_COLOR\_fake);



I ens dona el següent graf:



El graf ens mostra que les tasques no poden ser paralelitzades ja que depenen una de l’altre, això també es pot veure al T1 i al Tinf, que al ser quasi iguals, dona un paral·lelisme de 1.

I per últim ho fem amb -h

./run-tareador.sh mandel-seq-iter-tar -h

per generar l’histogram.

Aquí també hem canviat el codi.

Hem afegit cada cop que s'utilitza la variable histogram.

Per exemple:

if (output2histogram)

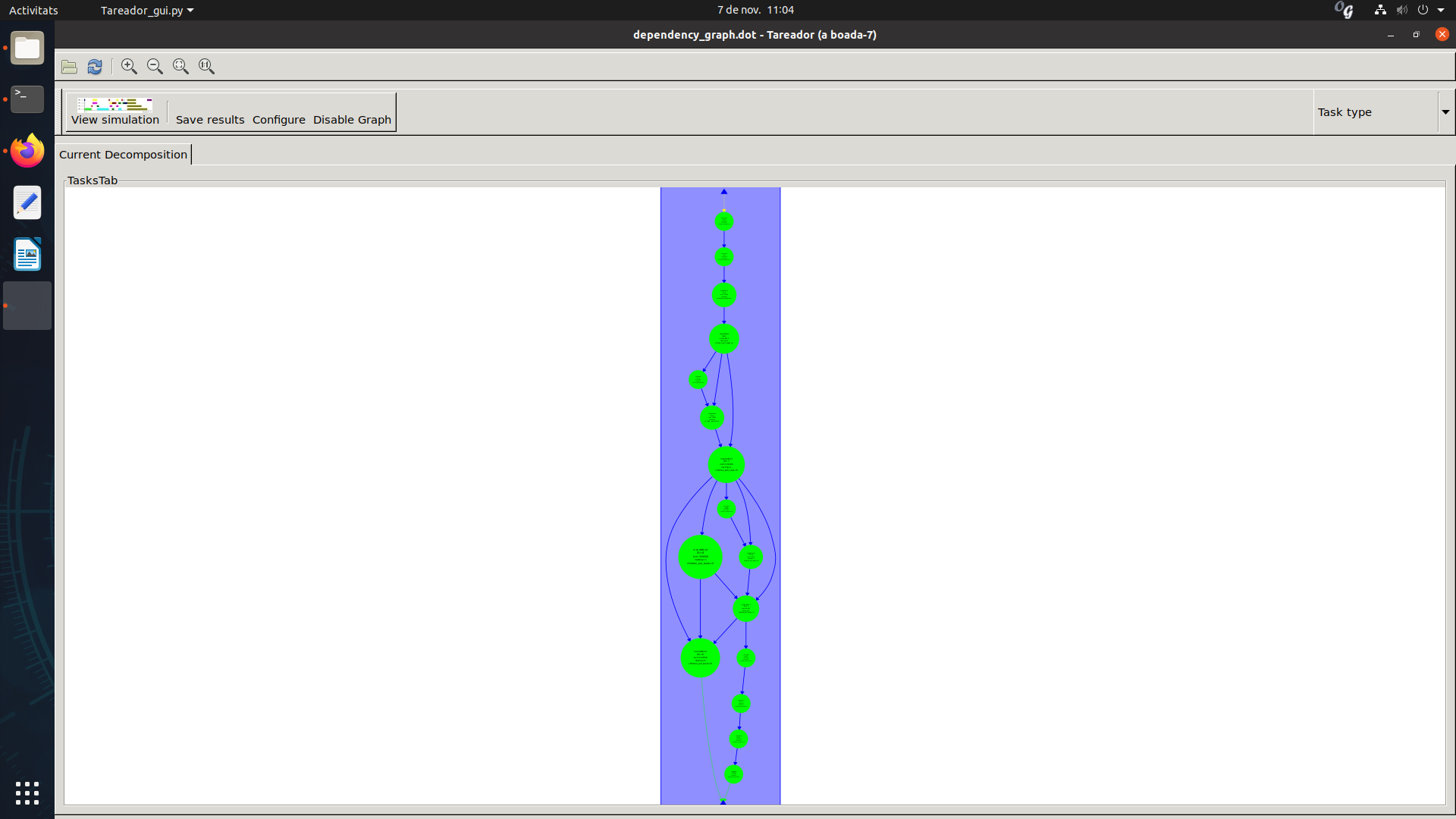
tareador\_disable\_object(histogram);

histogram[M[py][px] - 1]++;

tareador\_enable\_object(histogram);



I ens surt el següent graf.



En aquest graf es milloren una mica les dependències i el paral·lelisme millora una mica (1,072), encara que segueix sent molt petit perque només dues tasques es poden paralelitzar.

Aquests canvis que hem fet als codis han sigut per resoldre dependencies que tenien les variables alhora de fer les seves funcions (generar la imatge de mandelbrot o l’histograma).

Ara ens demanen que fem una paralelització “Fine-Grain”.

Per fer això modifiquem el codi fent que siguin tasques el comprovar els bordes verticals i horitzontals afegint

tareador\_start\_task("horitzontals");

for (int px = x; px < x + TILE; px++) {

M[y][px] = pixel\_dwell(COLS, ROWS, CminR, CminI, CmaxR, CmaxI, px, y, scale\_real, scale\_imag, maxiter);

M[y + TILE - 1][px] = pixel\_dwell(COLS, ROWS, CminR, CminI, CmaxR, CmaxI, px, y + TILE - 1, scale\_real, scale\_imag, maxiter);

equal = equal & (M[y][x] == M[y][px]);

equal = equal & (M[y][x] == M[y + TILE - 1][px]);

}

tareador\_end\_task("horitzontals");



I el mateix per al bucle dels bordes verticals.

A més també hem de fer tasques el condicional if-else enter, i dins d’aquest condicional el cada bucle de “fill” tots amb el mateix valor i cada bucle de py de computar a “tile”.

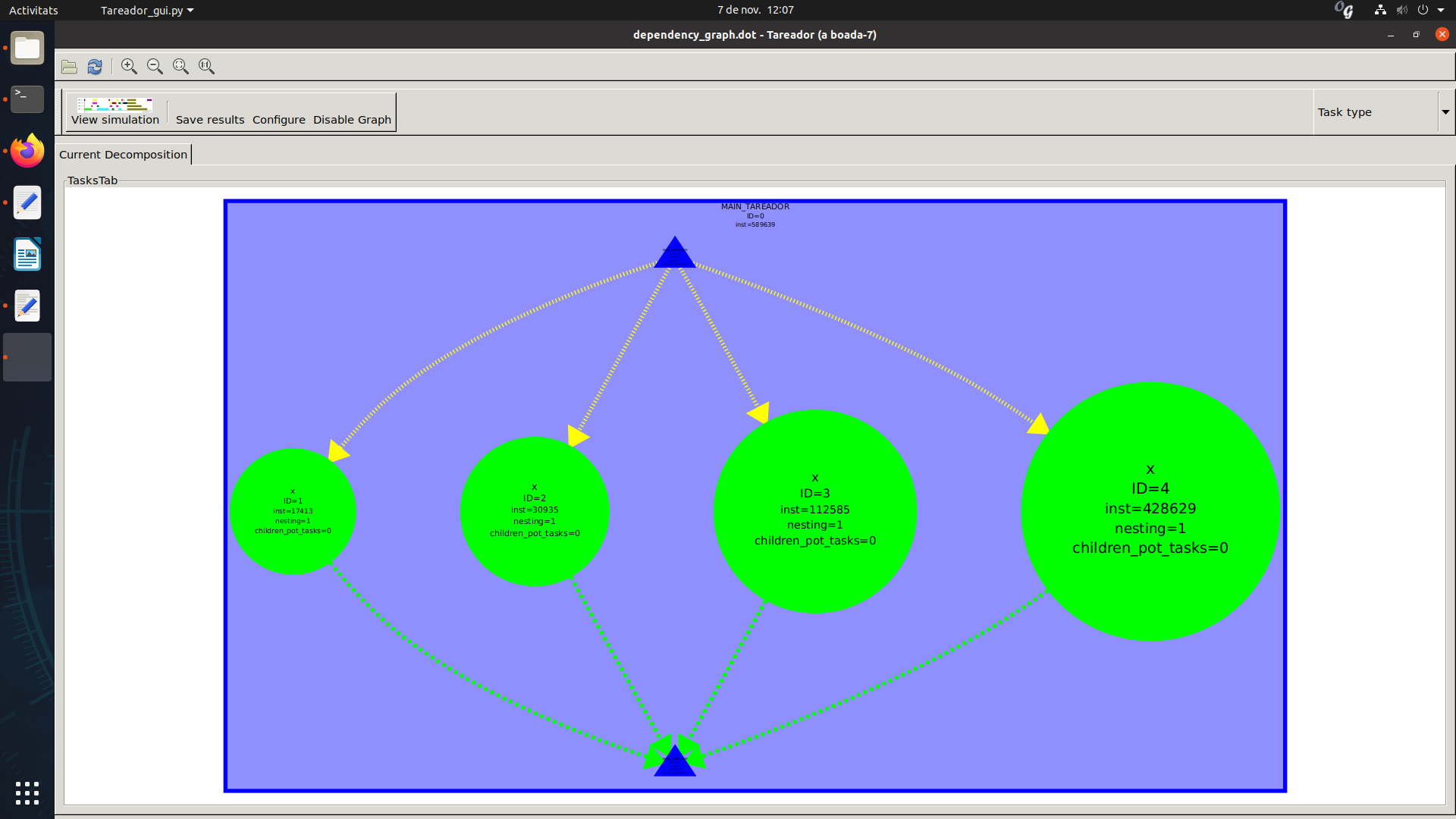
Fent aquests canvis ens surt un graf així:  
  


I per últim, ens demanen que apliquem una estratègia de paral·lelització que es dedica a processar una columna de “tiles”.

Per fer això hem fet que cada iteració de px sigui una tasca.

Ho hem fet afegint .

I ens ha sortit aquest graf:



**Recursiu:**

Leaf Recursive Task: el que volem en aquest primer codi es que les tasques es creïn només a les fulles, on la subdivisió ja no és possible perquè la grandària del tile ha aconseguit el límit (TILE), per això implementem tareador\_start\_task("Leaf");

Quan s’inicia el cas base, que es quan estem a una fulla de l’arbre i l’acabem, en el final d’aquest amb tareador\_end\_task("Leaf");

if (NCols <= TILE) {

tareador\_start\_task("Leaf");

// computation of tile: base case

for (int py = y; py < y + NRows; py++)

for (int px = x; px < x + NCols; px++) {

M[py][px] = pixel\_dwell(COLS, ROWS, CminR, CminI, CmaxR, CmaxI, px, py, scale\_real, scale\_imag, maxiter);

if (output2histogram)

histogram[M[py][px] - 1]++;

if (output2display) {

/\* Scale color and display point \*/

tareador\_disable\_object(display);

long color = (long)((M[py][px] - 1) \* scale\_color) + min\_color;

if (setup\_return == EXIT\_SUCCESS) {

XSetForeground(display, gc, color);

XDrawPoint(display, win, gc, px, py);

}

tareador\_enable\_object(display);

}

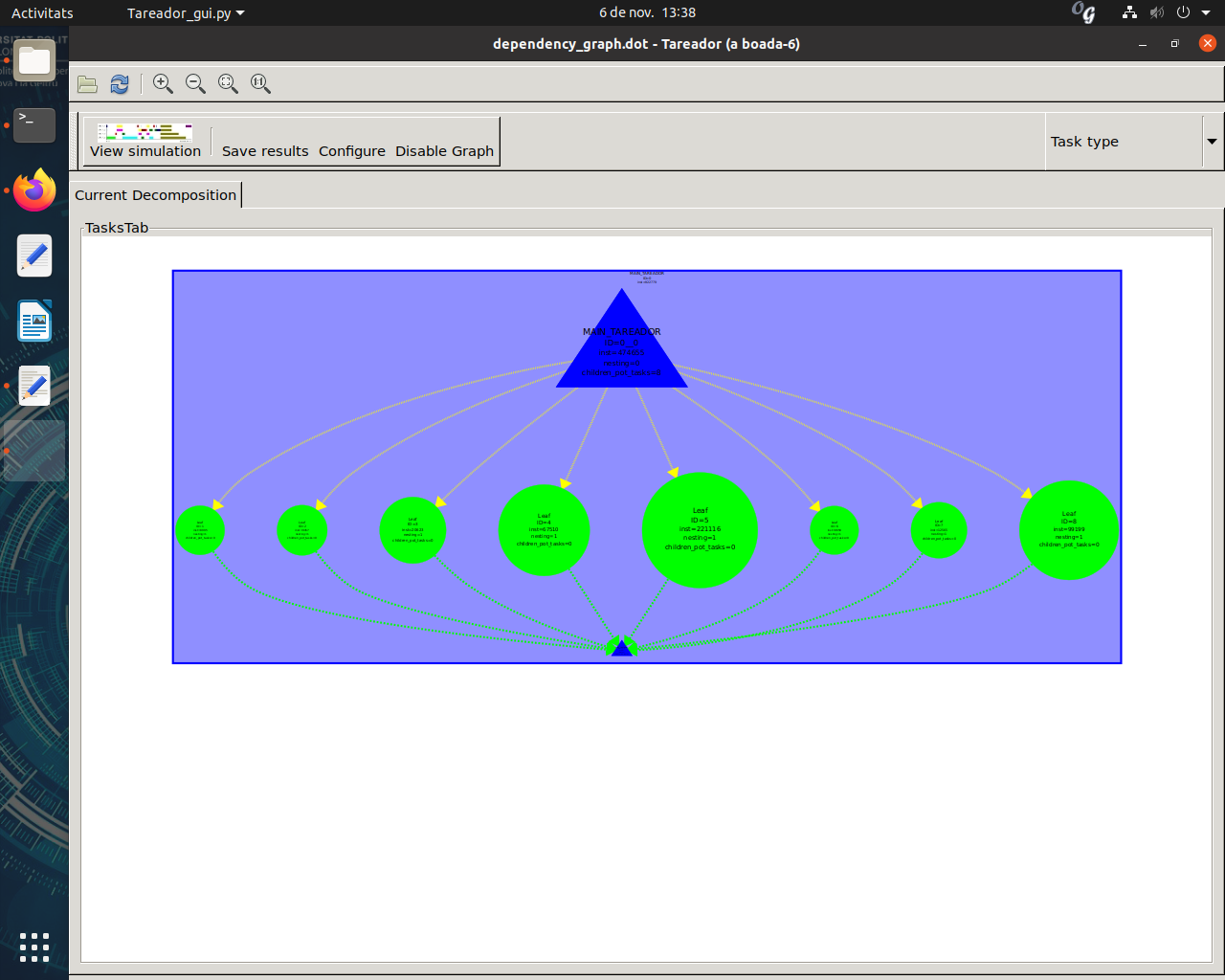
}

tareador\_end\_task("Leaf");

}

En aquest codi hem afegit ademés tareador\_disable\_object(&M); al inici del codi i cada cop que s’utiltzaba la variable display, amb lo cual reduïm les dependències entre tasques, facilitant que puguin executar-se en paral·lel sense esperar que s'alliberin altres recursos

I ens dona el següent graf:



Ara, per últim, hem fet el codi complet i li hem afegit tareador\_start\_task("fill"); dins del if on s’emplena completament un tile en la matriu amb el mateix valor si compleix la condició: if (equal) i al final d'aquest, hem acabat acabat la tasca tareador\_end\_task("fill");

if (equal) {

tareador\_start\_task("fill");

long color = (long)((M[y][x] - 1) \* scale\_color) + min\_color;

if (output2histogram)

histogram[M[y][x] - 1]+= (NRows\*NCols);

for (int py = y; py < y + NRows; py++)

for (int px = x; px < x + NCols; px++) {

M[py][px] = M[y][x];

if (output2display) {

tareador\_disable\_object(display);

/\* Scale color and display point \*/

if (setup\_return == EXIT\_SUCCESS) {

XSetForeground(display, gc, color);

XDrawPoint(display, win, gc, px, py);

}

tareador\_disable\_object(display);

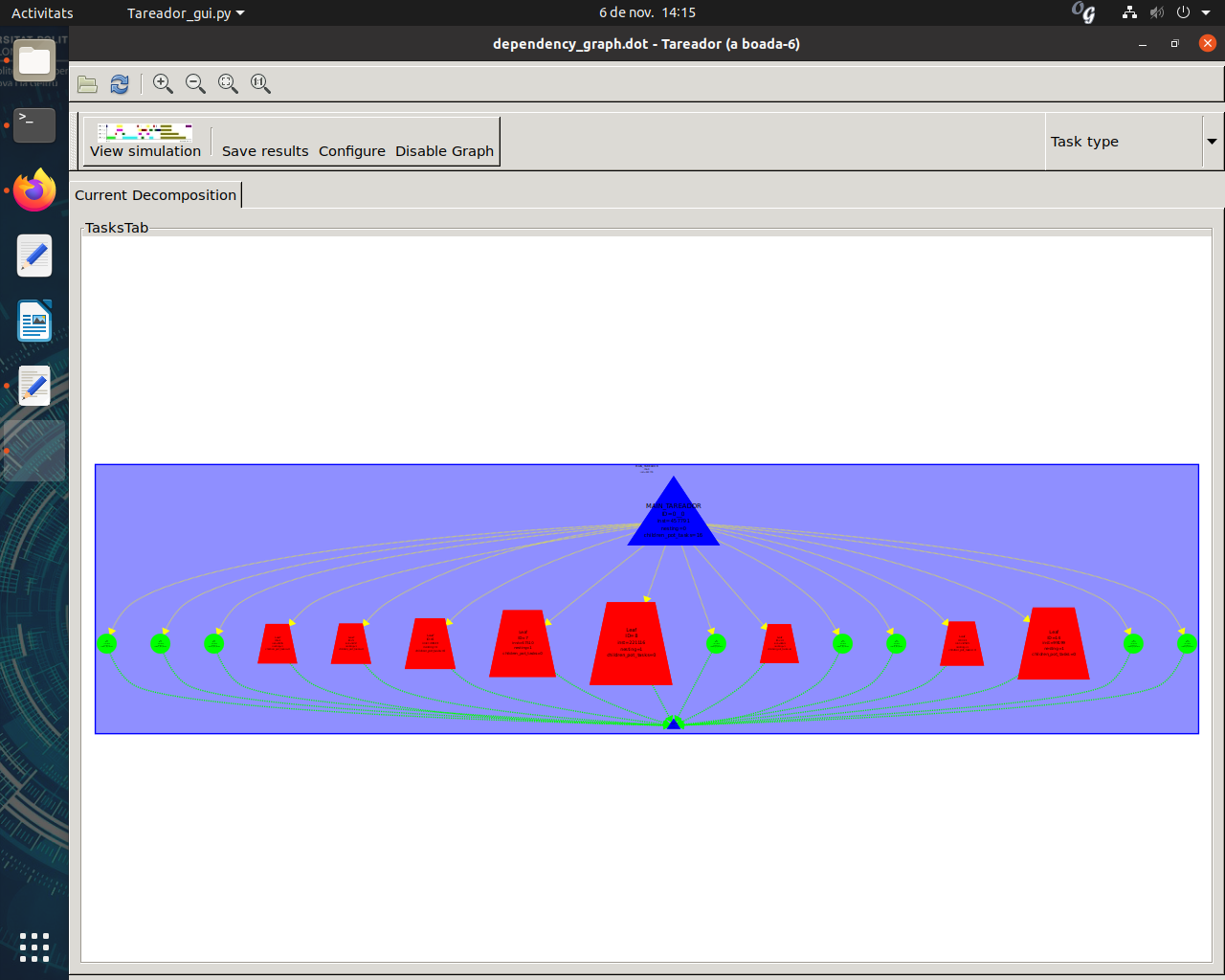
}

}

tareador\_end\_task("fill");

}

i el graf que obtenim és el següent:



Tree Recursive Task el que volem en aquest segon codi es que les tasques es creïn cada cop que fem la recursivitat és per això que creem les següents tasques en el codi:

if (NRows > TILE) {

tareador\_start\_task("Tree1");

mandel\_tiled\_rec(M, NRows / 2, NCols / 2, start\_fil, start\_col, CminR, CminI, CmaxR, CmaxI, scale\_real, scale\_imag, maxiter);

tareador\_end\_task("Tree1");

tareador\_start\_task("Tree2");

mandel\_tiled\_rec(M, NRows / 2, NCols / 2, start\_fil, start\_col + NCols / 2, CminR, CminI, CmaxR, CmaxI, scale\_real, scale\_imag, maxiter);

tareador\_end\_task("Tree2");

tareador\_start\_task("Tree3");

mandel\_tiled\_rec(M, NRows / 2, NCols / 2, start\_fil + NRows / 2, start\_col, CminR, CminI, CmaxR, CmaxI, scale\_real, scale\_imag, maxiter);

tareador\_end\_task("Tree3");

tareador\_start\_task("Tree4");

mandel\_tiled\_rec(M, NRows / 2, NCols / 2, start\_fil + NRows / 2, start\_col + NCols / 2, CminR, CminI, CmaxR, CmaxI, scale\_real, scale\_imag, maxiter);

tareador\_end\_task("Tree4");

} else {

tareador\_start\_task("Tree5");

mandel\_tiled\_rec(M, NRows, NCols / 2, start\_fil, start\_col, CminR, CminI, CmaxR, CmaxI, scale\_real, scale\_imag, maxiter);

tareador\_end\_task("Tree5");

tareador\_start\_task("Tree6");

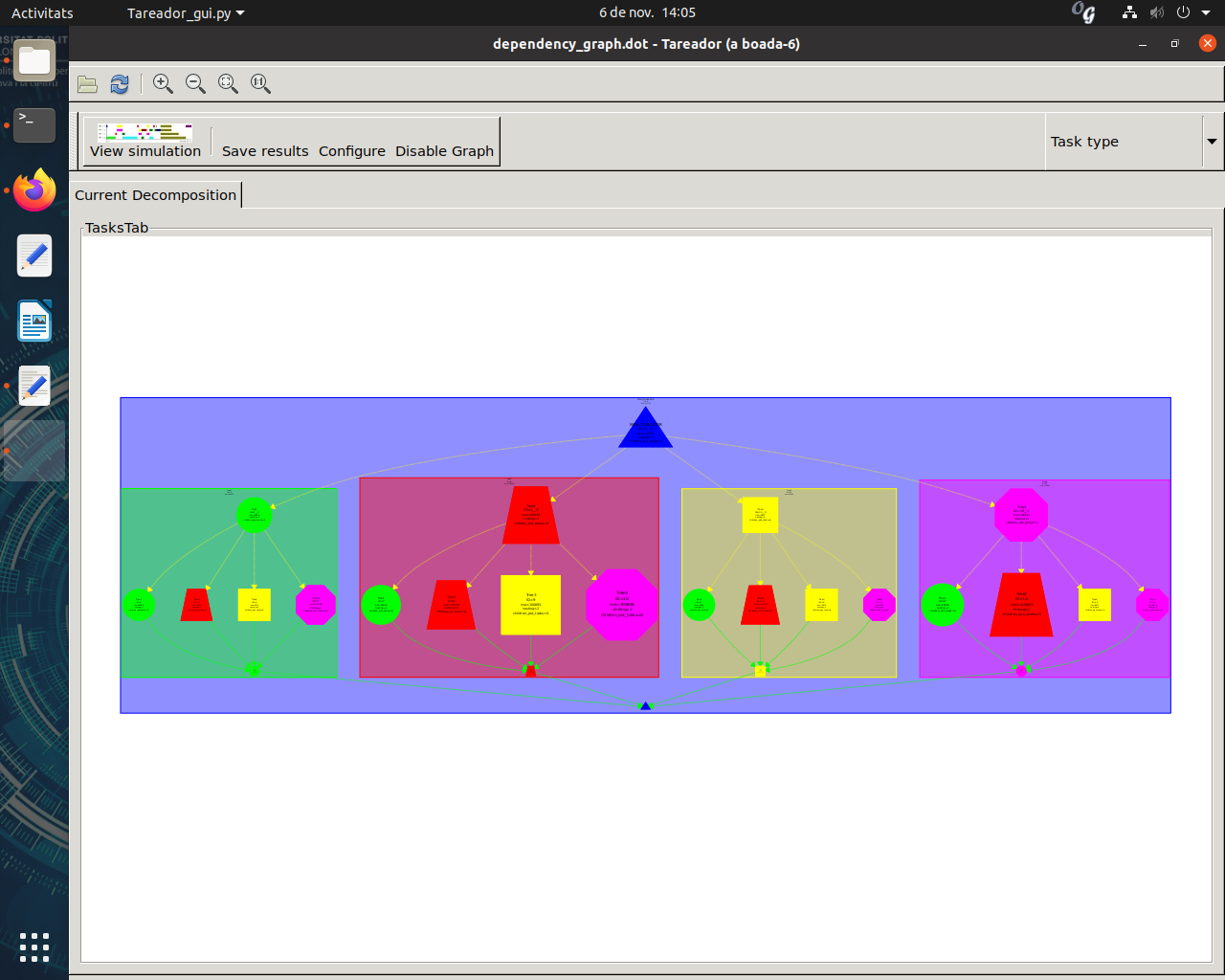
mandel\_tiled\_rec(M, NRows, NCols / 2, start\_fil, start\_col + NCols / 2, CminR, CminI, CmaxR, CmaxI, scale\_real, scale\_imag, maxiter);

tareador\_end\_task("Tree6");

}



i obtenim el següent graf:



Ara implementem tasques quan es comprova els bordes horitzontals i verticals:

tareador\_start\_task("horitzontals");tareador\_end\_task("horitzontals");

tareador\_start\_task("verticals");tareador\_end\_task("verticals");

tareador\_start\_task("horitzontals");

for (int px = x; px < x + NCols; px++) {

M[y][px] = pixel\_dwell(COLS, ROWS, CminR, CminI, CmaxR, CmaxI, px, y, scale\_real, scale\_imag, maxiter);

M[y + NRows - 1][px] = pixel\_dwell(COLS, ROWS, CminR, CminI, CmaxR, CmaxI, px, y + NRows - 1, scale\_real, scale\_imag, maxiter);

equal = equal & (M[y][x] == M[y][px]);

equal = equal & (M[y][x] == M[y + NRows - 1][px]);

}

tareador\_end\_task("horitzontals");

//check vertical borders

tareador\_start\_task("verticals");

for (int py = y; py < y + NRows; py++) {

M[py][x] = pixel\_dwell(COLS, ROWS, CminR, CminI, CmaxR, CmaxI, x, py, scale\_real, scale\_imag, maxiter);

M[py][x + NCols - 1] = pixel\_dwell(COLS, ROWS, CminR, CminI, CmaxR, CmaxI, x + NCols - 1, py, scale\_real, scale\_imag, maxiter);

equal = equal & (M[y][x] == M[py][x]);

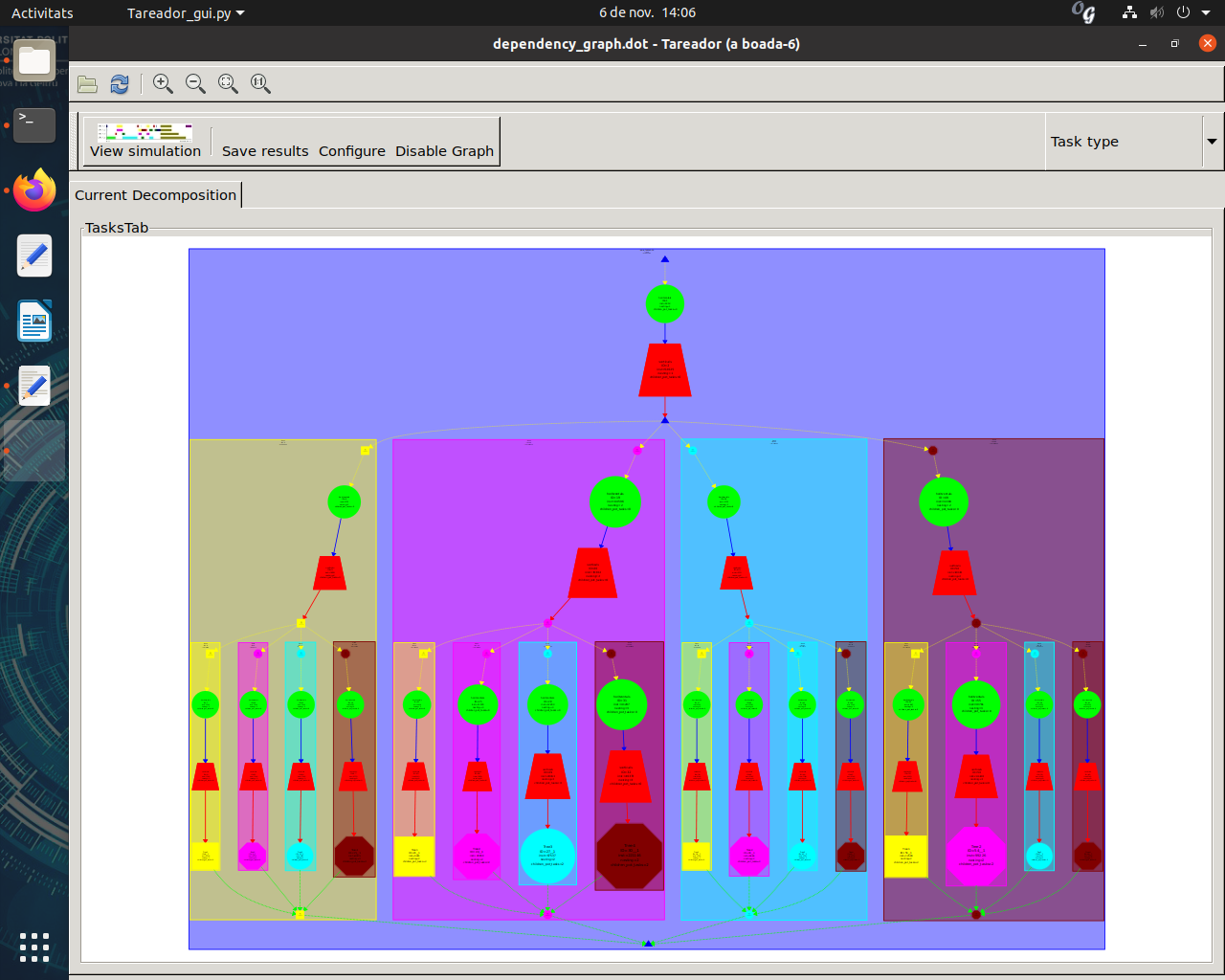
equal = equal & (M[y][x] == M[py][x + NCols - 1]);

}

tareador\_end\_task("verticals");



i ens surt aquest graf:



Per últim, afegim tareador\_start\_task("fill"); dins del if on s’emplena completament un tile en la matriu amb el mateix valor si compleix la condició: if (equal) i al final d'aquest, hem acabat acabat la tasca tareador\_end\_task("fill");

if (equal) {

tareador\_start\_task("fill");

long color = (long)((M[y][x] - 1) \* scale\_color) + min\_color;

if (output2histogram)

histogram[M[y][x] - 1]+= (NRows\*NCols);

for (int py = y; py < y + NRows; py++)

for (int px = x; px < x + NCols; px++) {

M[py][px] = M[y][x];

if (output2display) {

tareador\_disable\_object(display);

/\* Scale color and display point \*/

if (setup\_return == EXIT\_SUCCESS) {

XSetForeground(display, gc, color);

XDrawPoint(display, win, gc, px, py);

}

tareador\_disable\_object(display);

}

}

tareador\_end\_task("fill");

}

i aconseguim aquest graf final:

