

Tool for configuration of coordinated cache and bank coloring

Bjorn Andersson

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2021 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM21-0235

Background:

- * **About multicore.** Multicore processors are the norm today. This has been the case in desktop and laptop computers for 15 years. This has been the case for mobile computing for 10 years. In DoD, for non-safety-critical systems, multicore systems have started to be used. In DoD, for safety-critical systems, multicores (with all cores enabled) are currently not used but there is great interest in finding ways to do so.
- * **About multicore and timing.** Multicore processors tend to have shared memory (both DRAM memory shared between processor cores and shared physical address space between processor cores). There are also many resources that aim to speed up execution (e.g., cache) and there are resources for connecting processor cores to memory (e.g., memory bus). On a higher level, one can say that there are resources in the memory system that are shared between processor cores and hence they may be shared between processes/threads on different processor cores. For some of these resources, it holds that the resource can only be used by one thread/process at a time (e.g., memory bus). For other resources, it holds that the use of one resource by one thread causes another thread later to execute more slowly (e.g., cache eviction).
- * **Mechanism.** The SEI has developed a technique called coordinated cache and bank coloring. See N. Suzuki et al., “Coordinated bank and cache coloring for temporal protection of memory accesses,” ICSS 2013.
This technique involves changing the virtual memory manager in the operating system. It also involves deciding how the virtual memory manager should work; we call it configuration. This slide deck presents the tool for configuration.

* **The advantages of the mechanism.** For a mechanism, two properties are desirable:

1. Providing timing isolation with respect to both cache and bank.
2. Requiring only the use of the virtual memory system (i.e., not depend on instructions or features that are available in only few processors). This is important for systems that may have long life-time (decades).

As far as we know, coordinated cache and bank coloring is the only technique that offers these two properties. (there are other techniques for cache coloring but they do not achieve bank coloring; there are techniques for bank coloring but they do not achieve cache coloring; there are techniques that use cache locking but this requires special instructions).

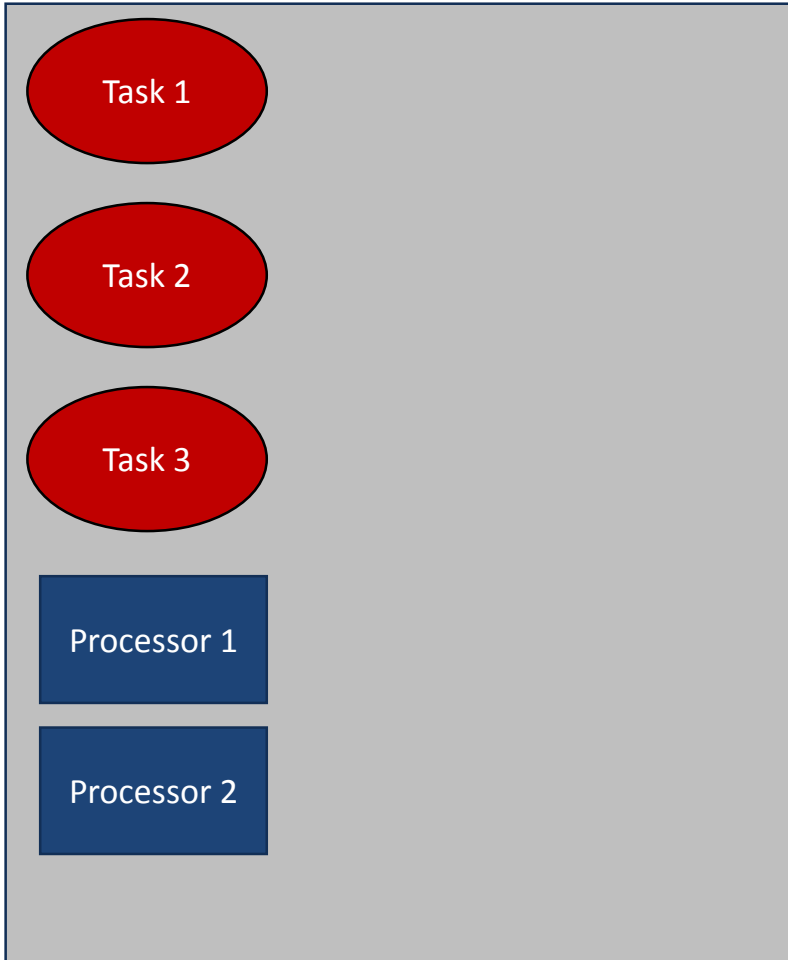
* **The disadvantages of the mechanism:**

1. Since it requires changing the behavior of the virtual memory system, it requires changing the operating system.
2. May require that at most a fraction of the available physical memory is used.

There are ways to alleviate these drawbacks. For example, for the former, it is sometimes possible to use kernel modules. For the latter, one can distinguish hard and soft real-time and take advantage of that.

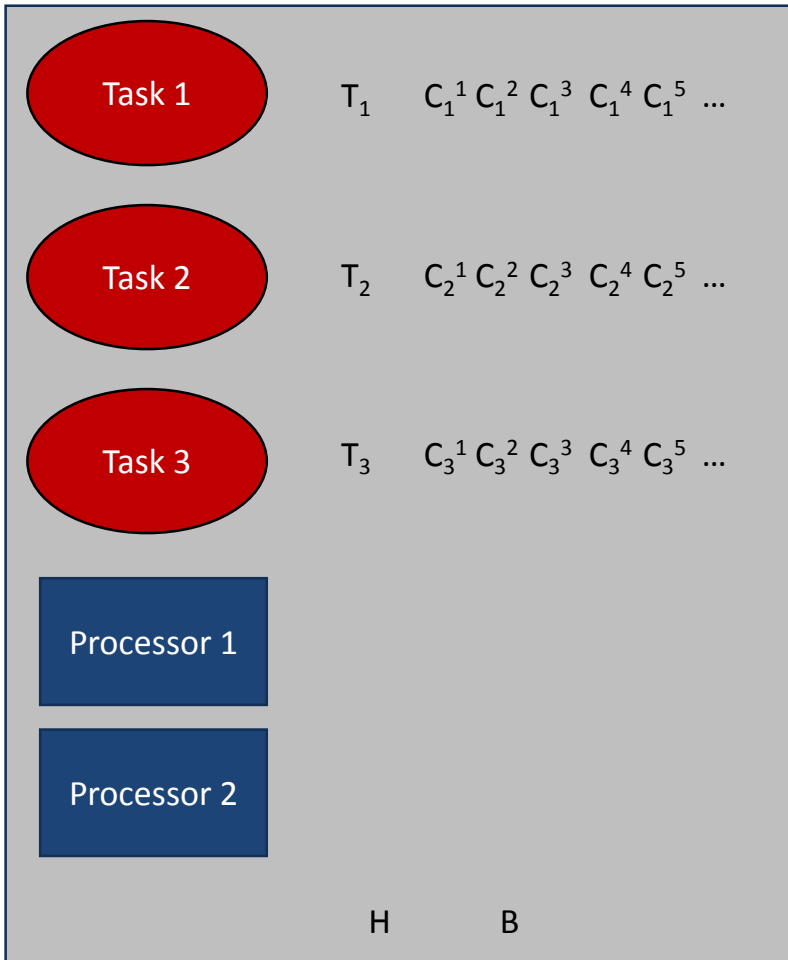
What the configuration tool does

Model of the system



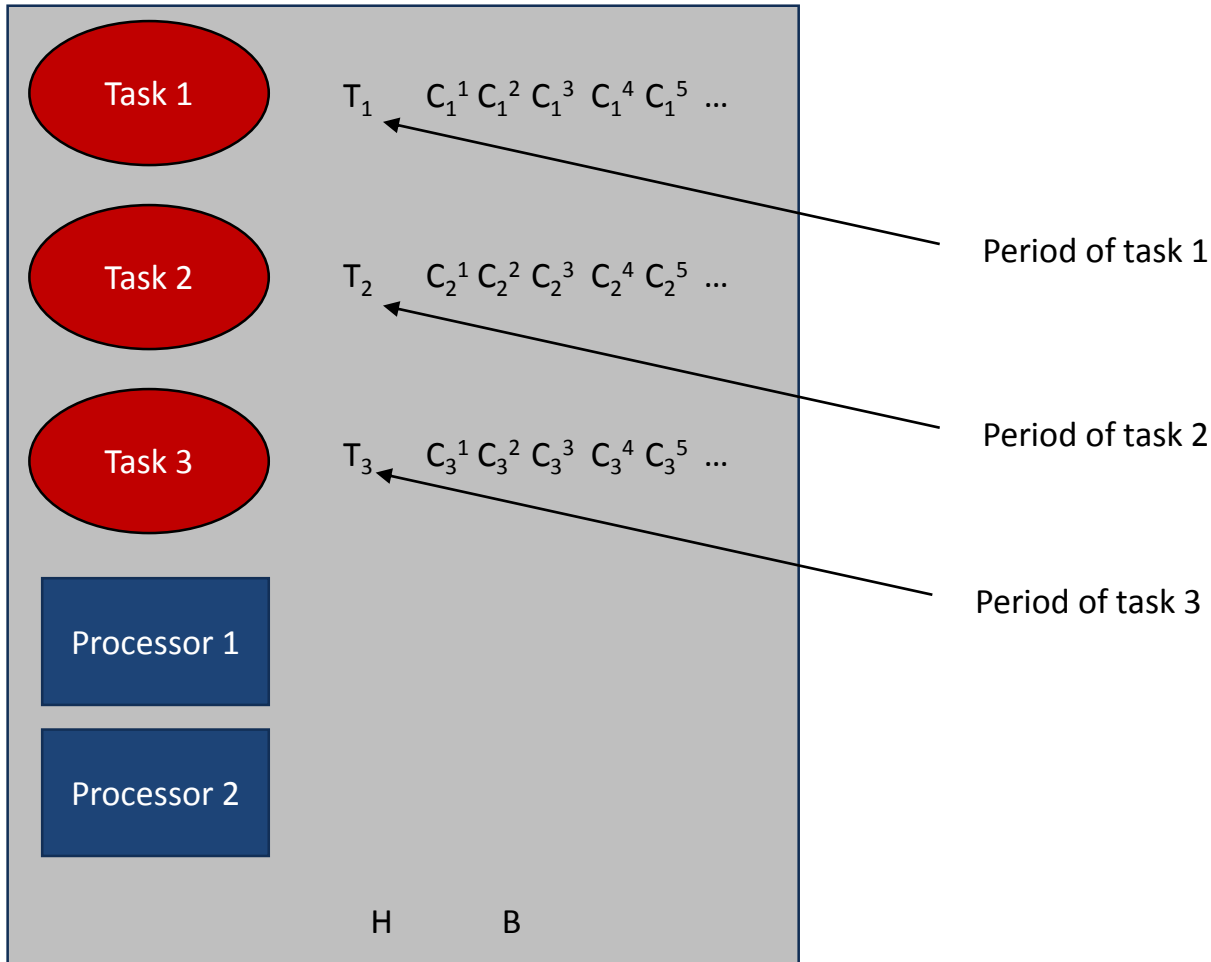
What the configuration tool does

Model of the system



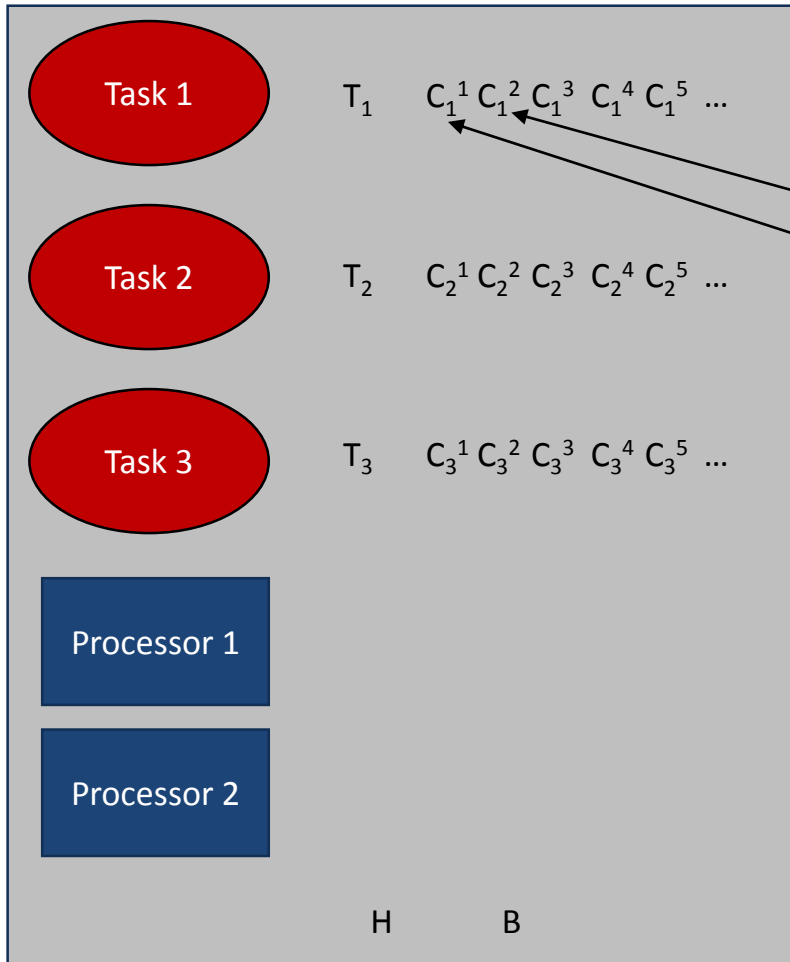
What the configuration tool does

Model of the system



What the configuration tool does

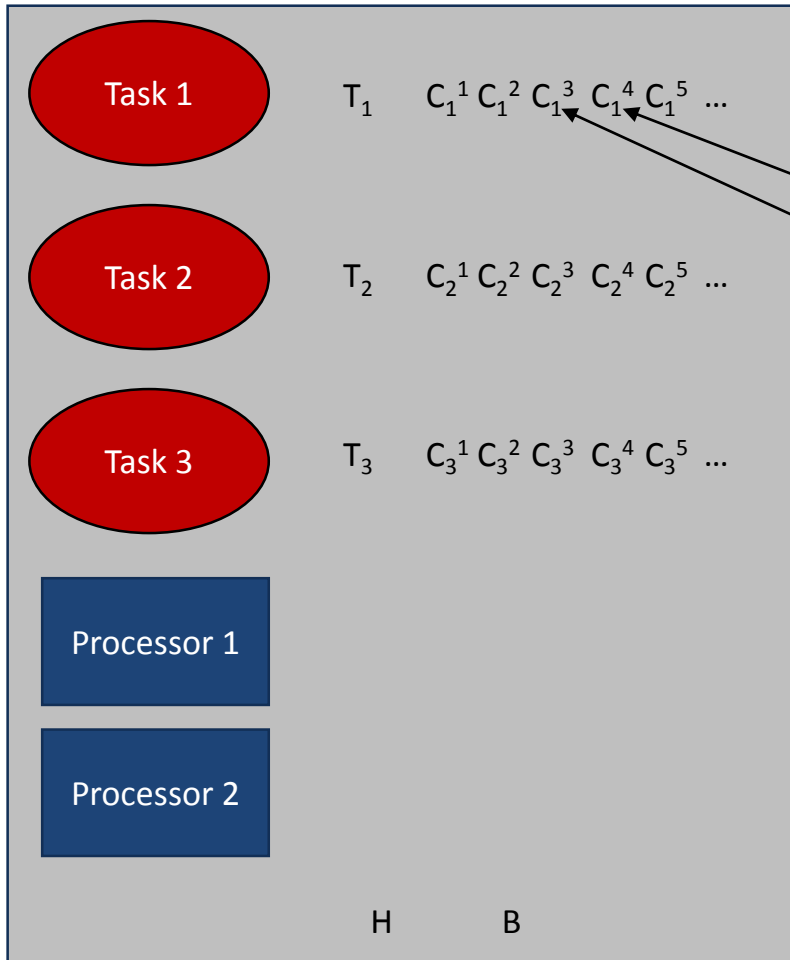
Model of the system



Execution time of task 1 if it is assigned 2 cache colors
Execution time of task 1 if it is assigned 1 cache color

What the configuration tool does

Model of the system

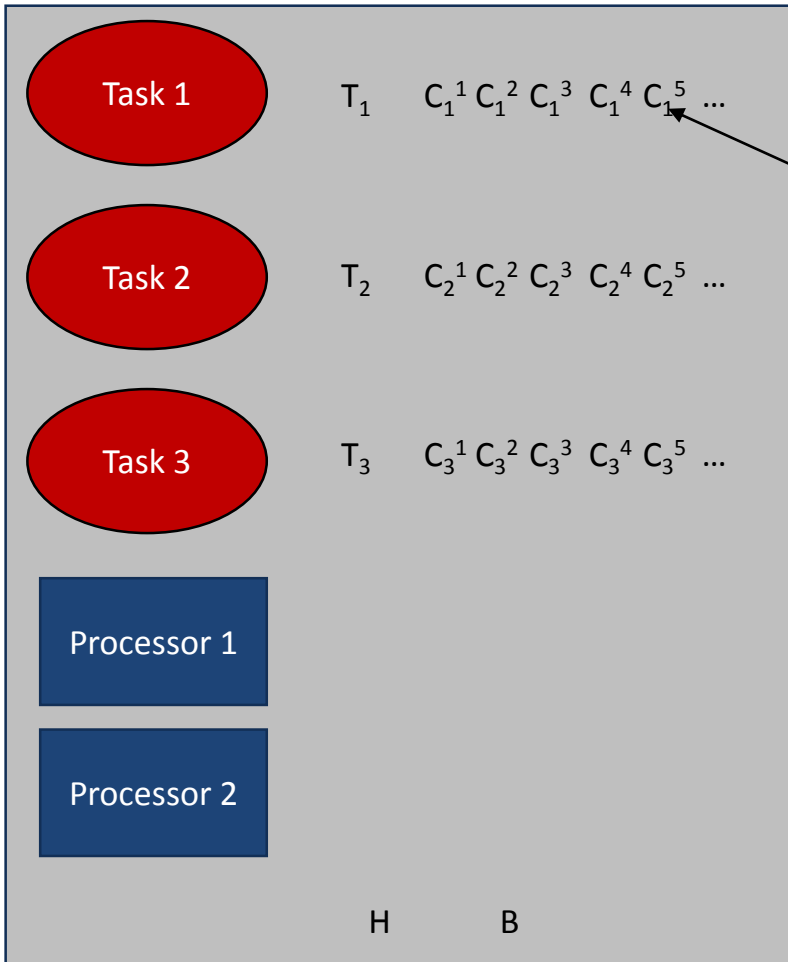


Execution time of task 1 if it is assigned 4 cache colors

Execution time of task 1 if it is assigned 3 cache colors

What the configuration tool does

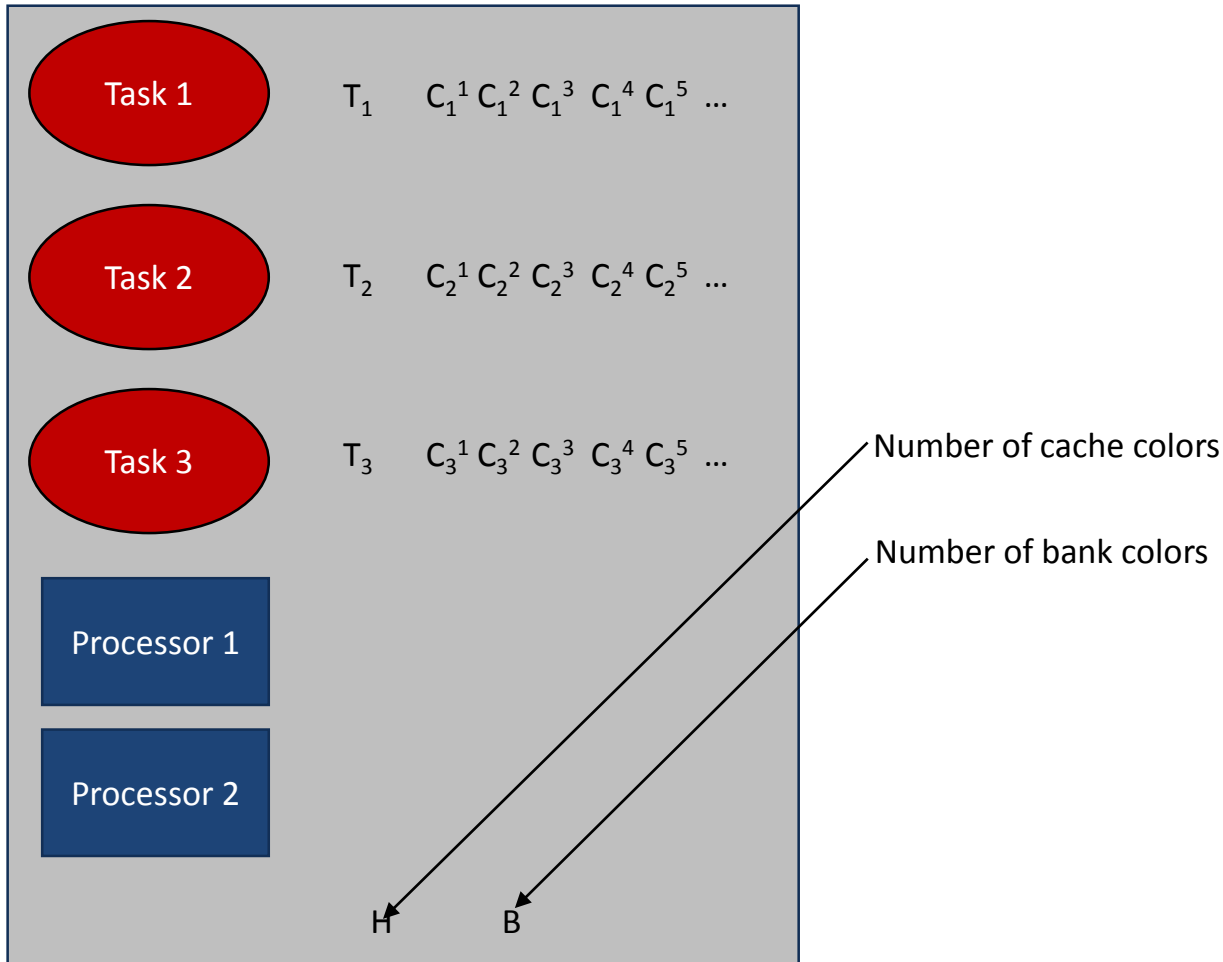
Model of the system



Execution time of task 1 if it is assigned 5 cache colors

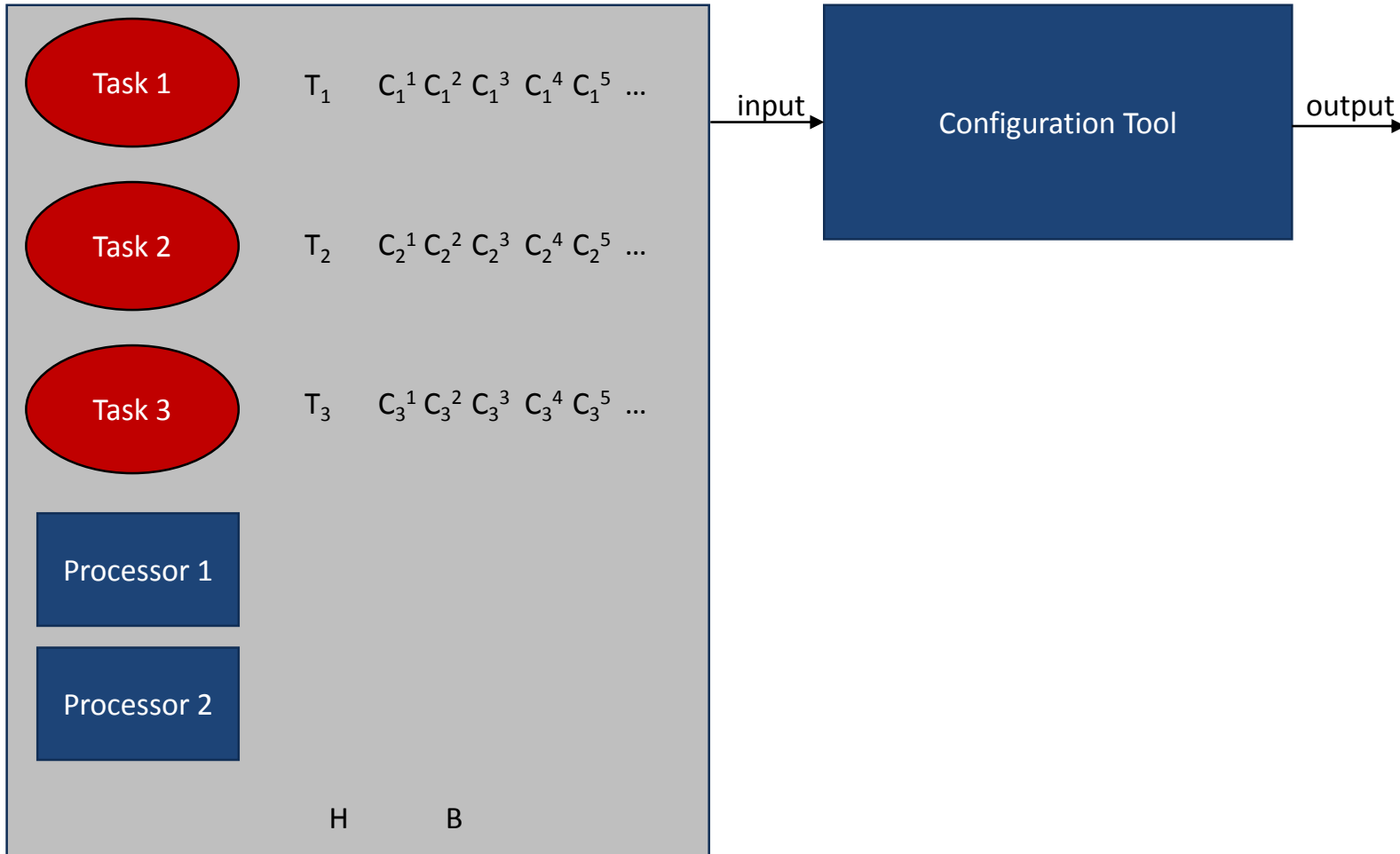
What the configuration tool does

Model of the system



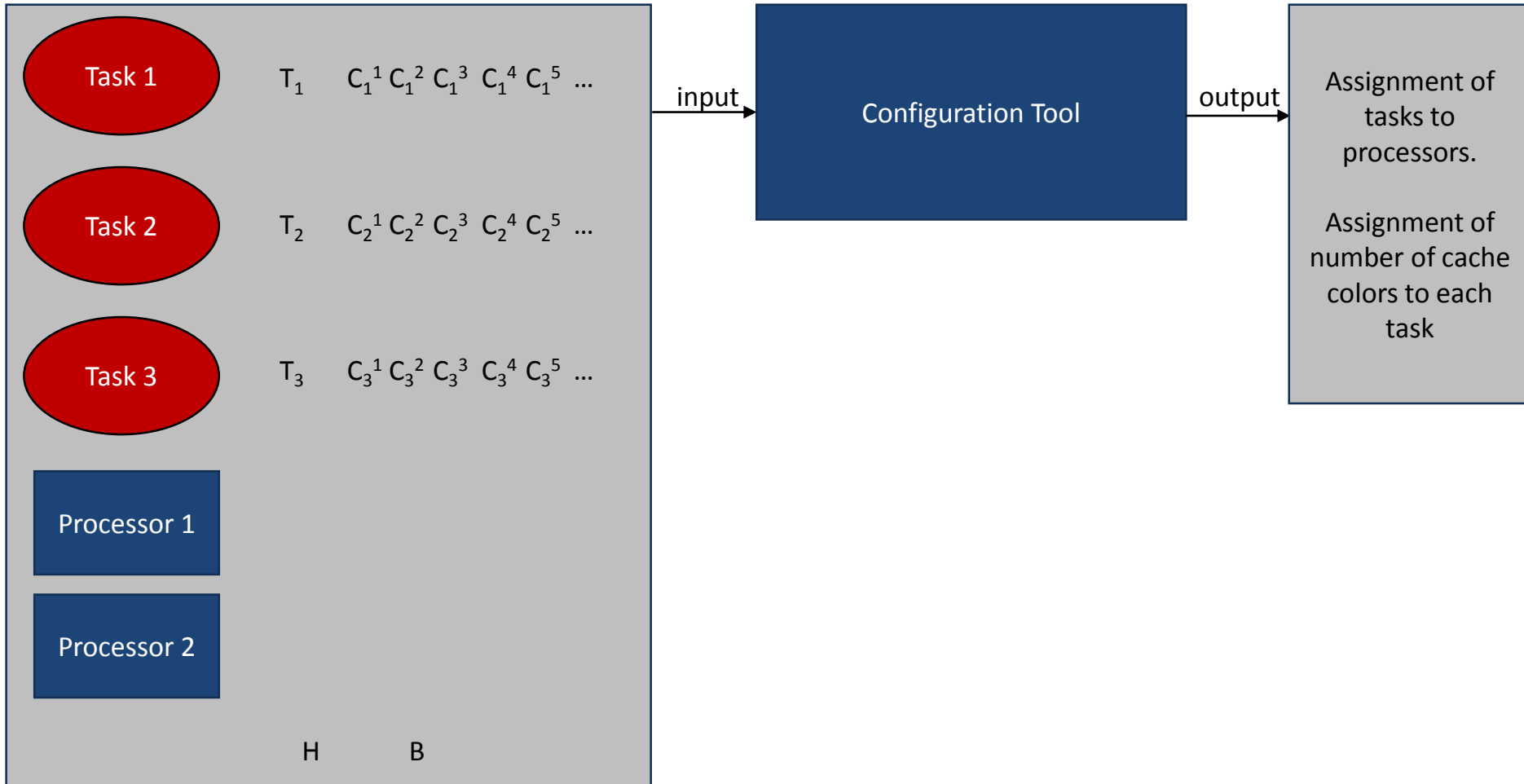
What the configuration tool does

Model of the system



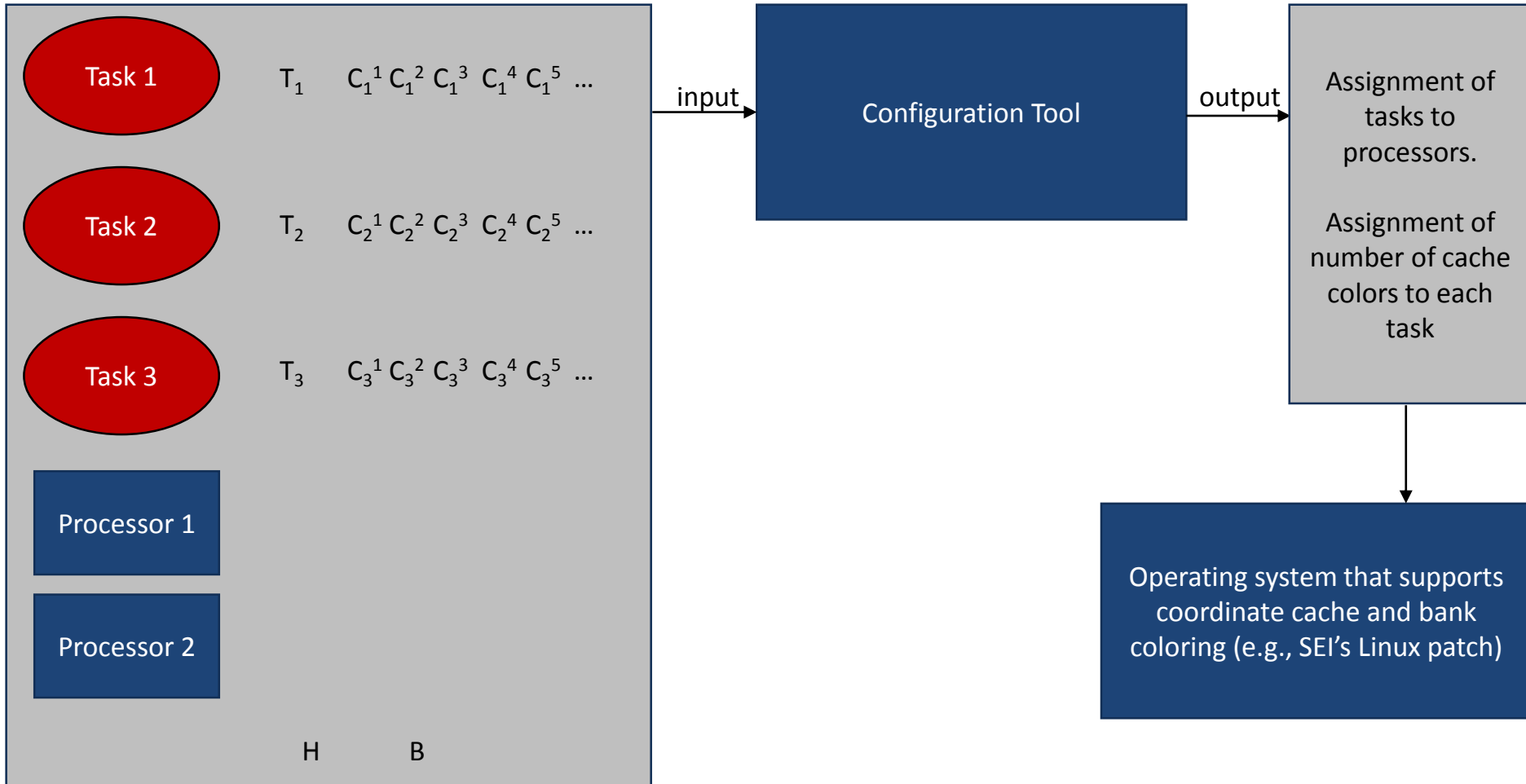
What the configuration tool does

Model of the system



What the configuration tool does

Model of the system



Understanding the constraints on configuration

Consider a computer system with 4 cache colors and 4 bank colors and two tasks τ_1 and τ_2 .
The physical memory consists of 16 cells.
Assume task 1 and task 2 are assigned to the same processor, processor 1.

		Cache color			
		0	1	2	3
Bank color	0				
	1				
	2				
	3				

Understanding the constraints on configuration

Consider a computer system with 4 cache colors and 4 bank colors and two tasks τ_1 and τ_2 .
The physical memory consists of 16 cells.
Assume task 1 and task 2 are assigned to the same processor, processor 1.

		Cache color			
		0	1	2	3
Bank color	0				
	1				
	2	Task 2	Task 1		Task 1
	3				

Understanding the constraints on configuration

Consider a computer system with 4 cache colors and 4 bank colors and two tasks τ_1 and τ_2 .
The physical memory consists of 16 cells.
Assume task 1 and task 2 are assigned to the same processor, processor 1.

		Cache color			
		0	1	2	3
Bank color	0				
	1				
	2	Task 2	Task 1		Task 1
	3				

Note that task 1 and task 2 do not share cache colors.

Understanding the constraints on configuration

Consider a computer system with 4 cache colors and 4 bank colors and two tasks τ_1 and τ_2 .
The physical memory consists of 16 cells.
Assume task 1 and task 2 are assigned to the same processor, processor 1.

		Cache color			
		0	1	2	3
Bank color	0				
	1				
	2	Task 2	Task 1		Task 1
	3				

Note that task 1 and task 2 shares bank colors.

But this is OK because they are assigned to the same processor.

Understanding the constraints on configuration

Consider a computer system with 4 cache colors and 4 bank colors and two tasks τ_1 and τ_2 .

The physical memory consists of 16 cells.

Assume task 1 and task 2 are assigned to different processors, processor 1 and processor 2 respectively.

		Cache color			
		0	1	2	3
Bank color	0				
	1				
	2				
	3				

Understanding the constraints on configuration

Consider a computer system with 4 cache colors and 4 bank colors and two tasks τ_1 and τ_2 .

The physical memory consists of 16 cells.

Assume task 1 and task 2 are assigned to different processors, processor 1 and processor 2 respectively.

		Cache color			
		0	1	2	3
Bank color	0				
	1	Task 2			
	2		Task 1		Task 1
	3				

Understanding the constraints on configuration

Consider a computer system with 4 cache colors and 4 bank colors and two tasks τ_1 and τ_2 .

The physical memory consists of 16 cells.

Assume task 1 and task 2 are assigned to different processors, processor 1 and processor 2 respectively.

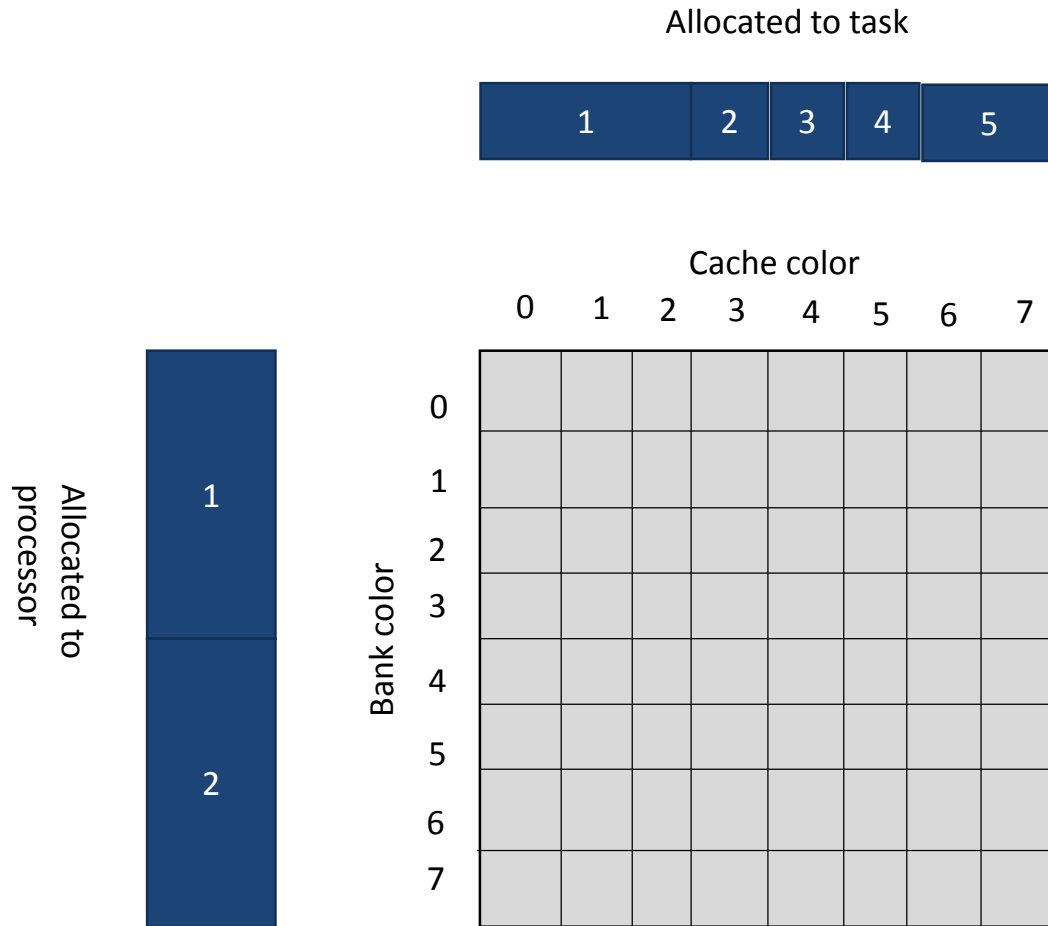
		Cache color			
		0	1	2	3
Bank color	0				
	1	Task 2			
	2		Task 1		Task 1
	3				

Note that task 1 and task 2 are assigned to different cache colors and bank colors.

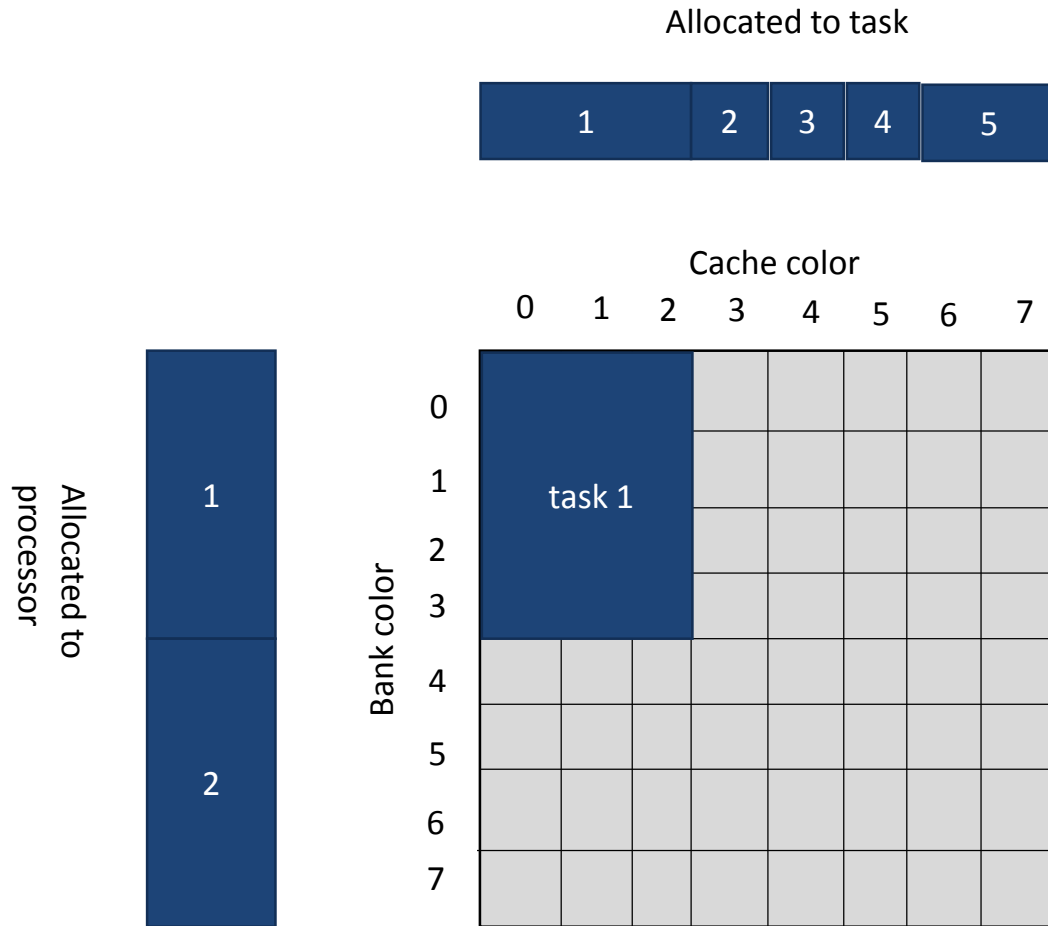
Since task 1 and task 2 are assigned to different processors, they must use different bank colors.

The figure on previous slide shows a concrete allocation. We will however create an abstract allocation which can then be converted to a concrete allocation. The next slide shows an abstract allocation.

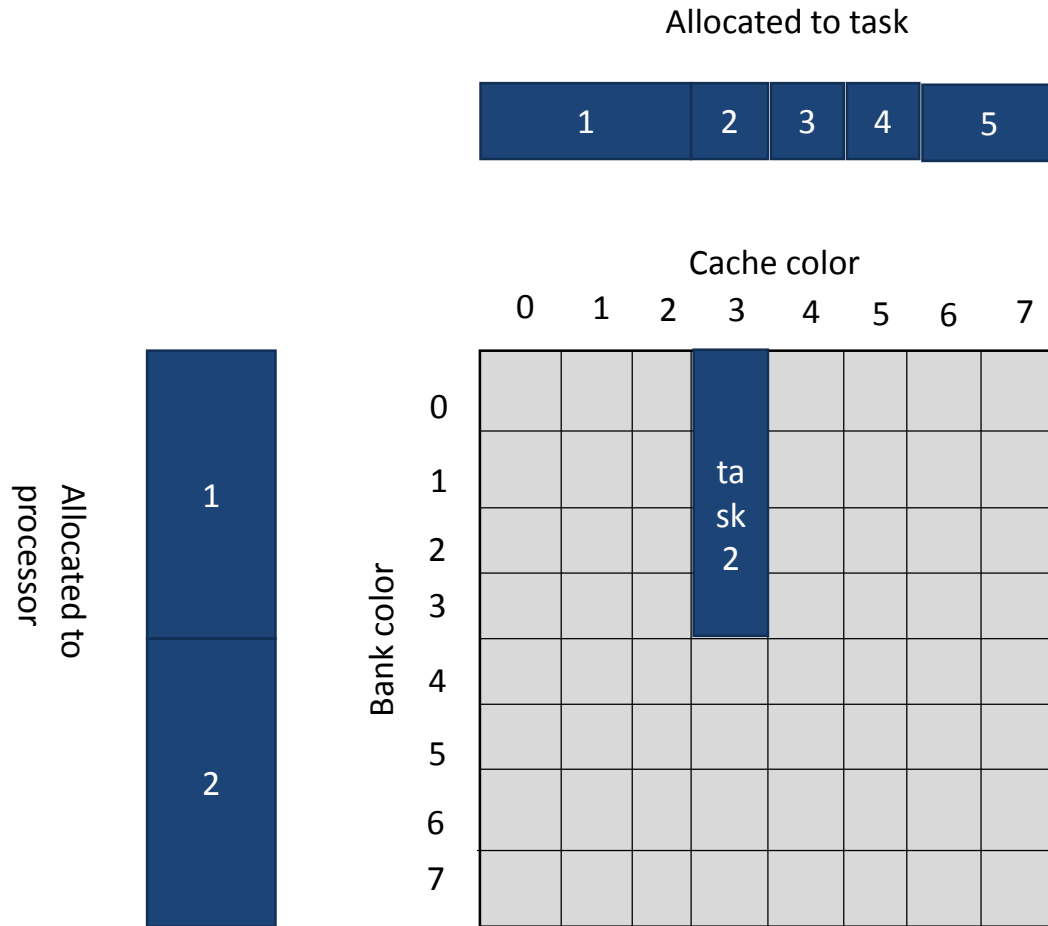
Assume task 1 and task 2 are assigned to processor 1 and task 3, task 4, and task 5 are assigned to processor 2.



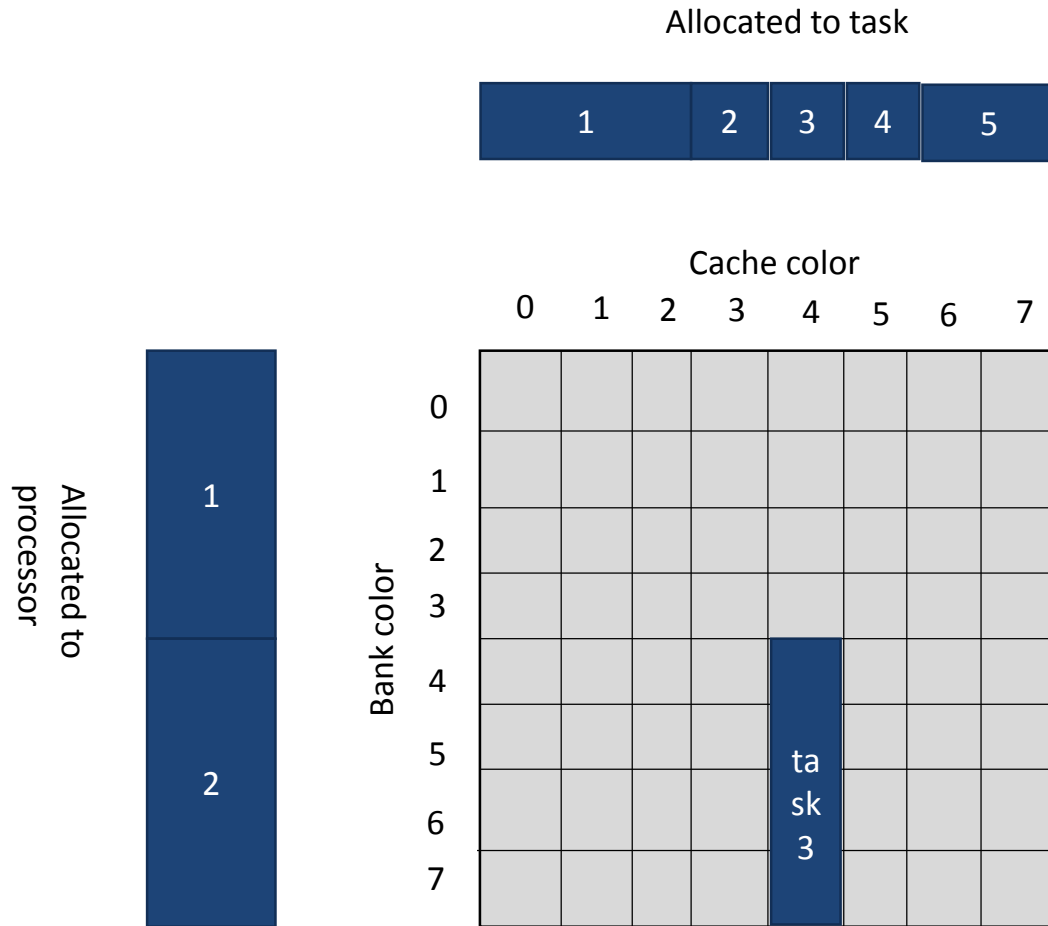
Assume task 1 and task 2 are assigned to processor 1 and task 3, task 4, and task 5 are assigned to processor 2.



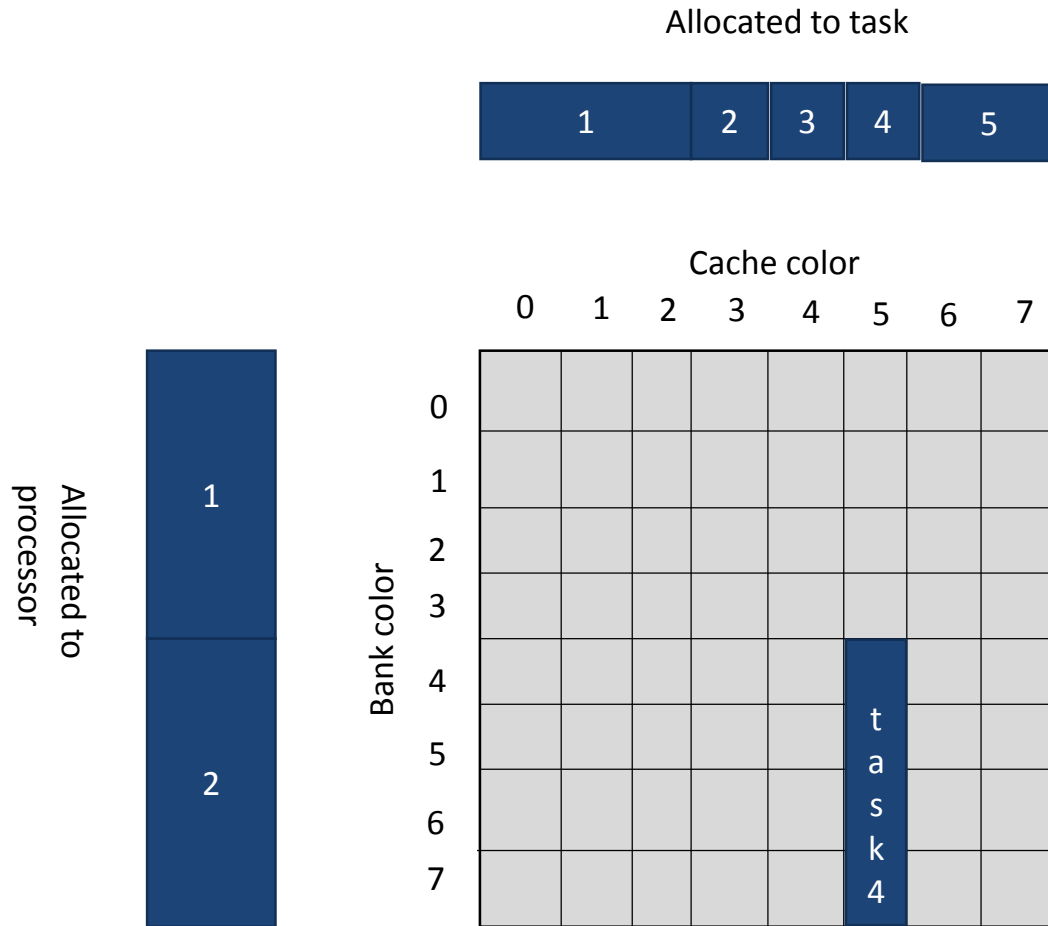
Assume task 1 and task 2 are assigned to processor 1 and task 3, task 4, and task 5 are assigned to processor 2.



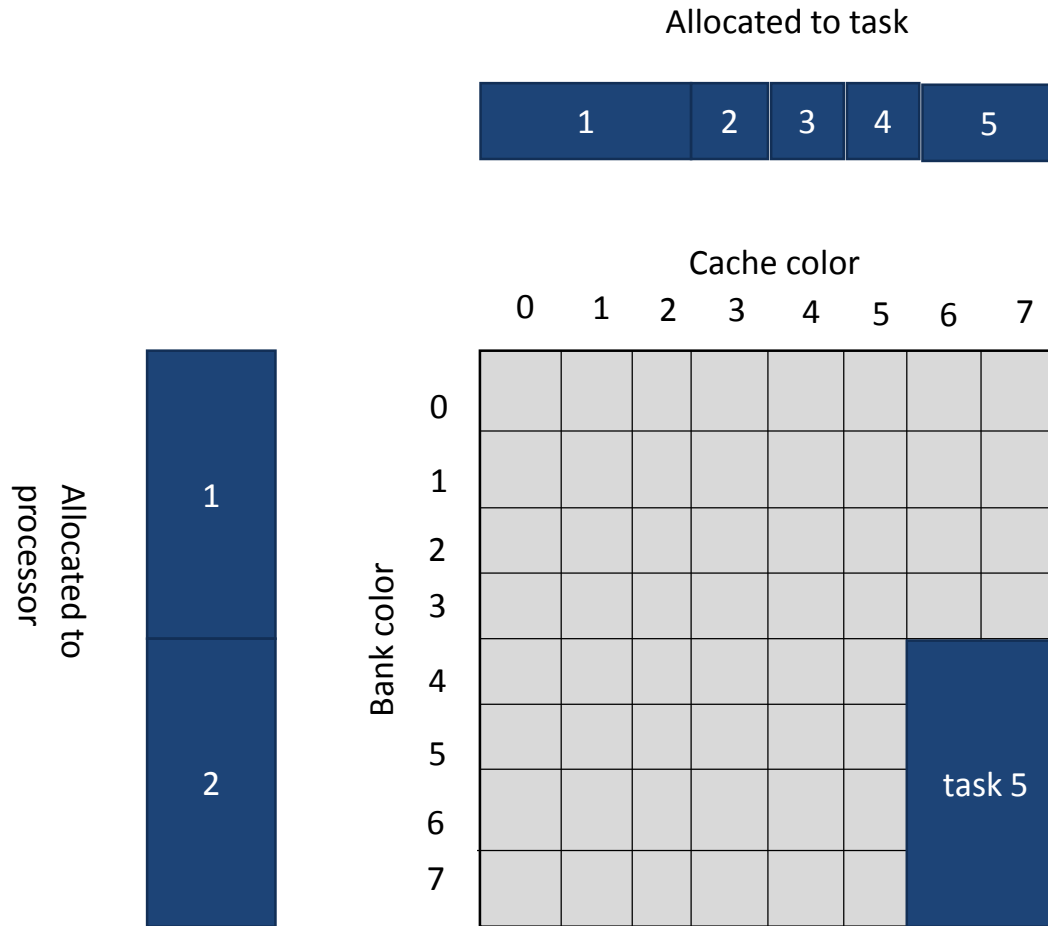
Assume task 1 and task 2 are assigned to processor 1 and task 3, task 4, and task 5 are assigned to processor 2.



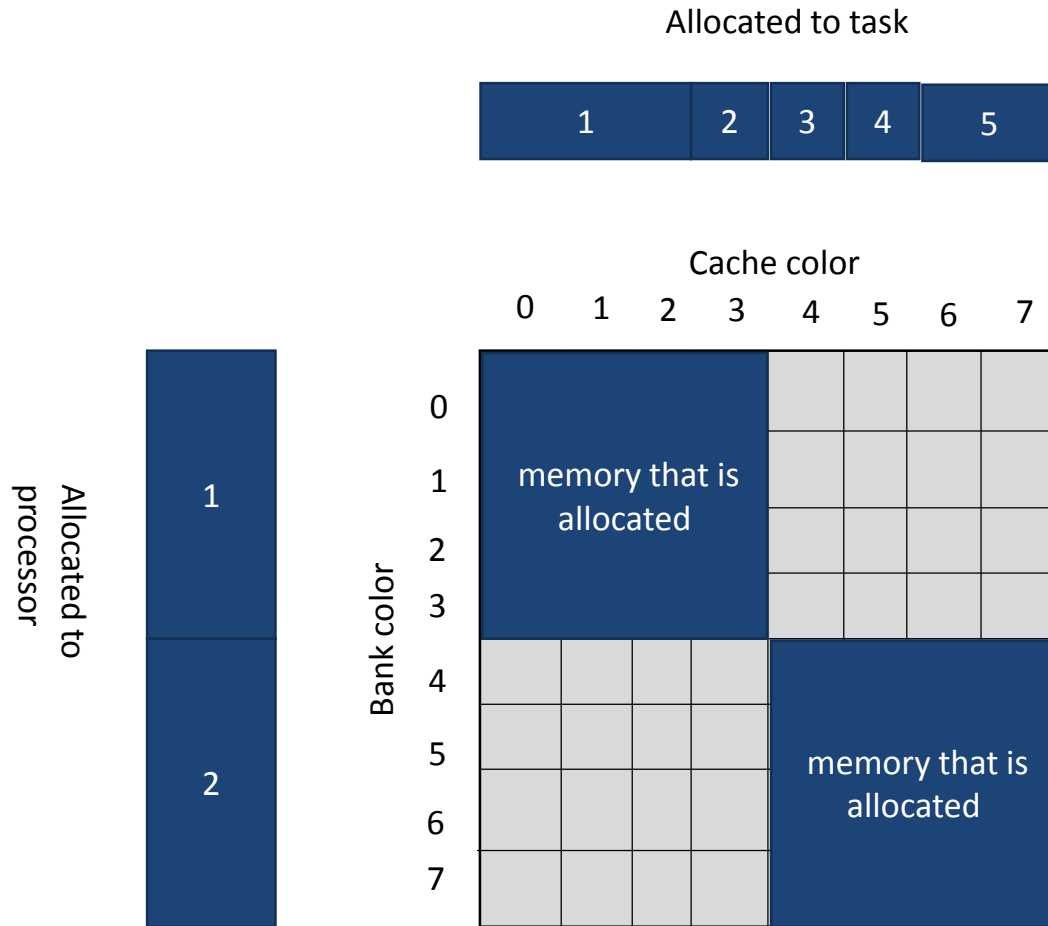
Assume task 1 and task 2 are assigned to processor 1 and task 3, task 4, and task 5 are assigned to processor 2.



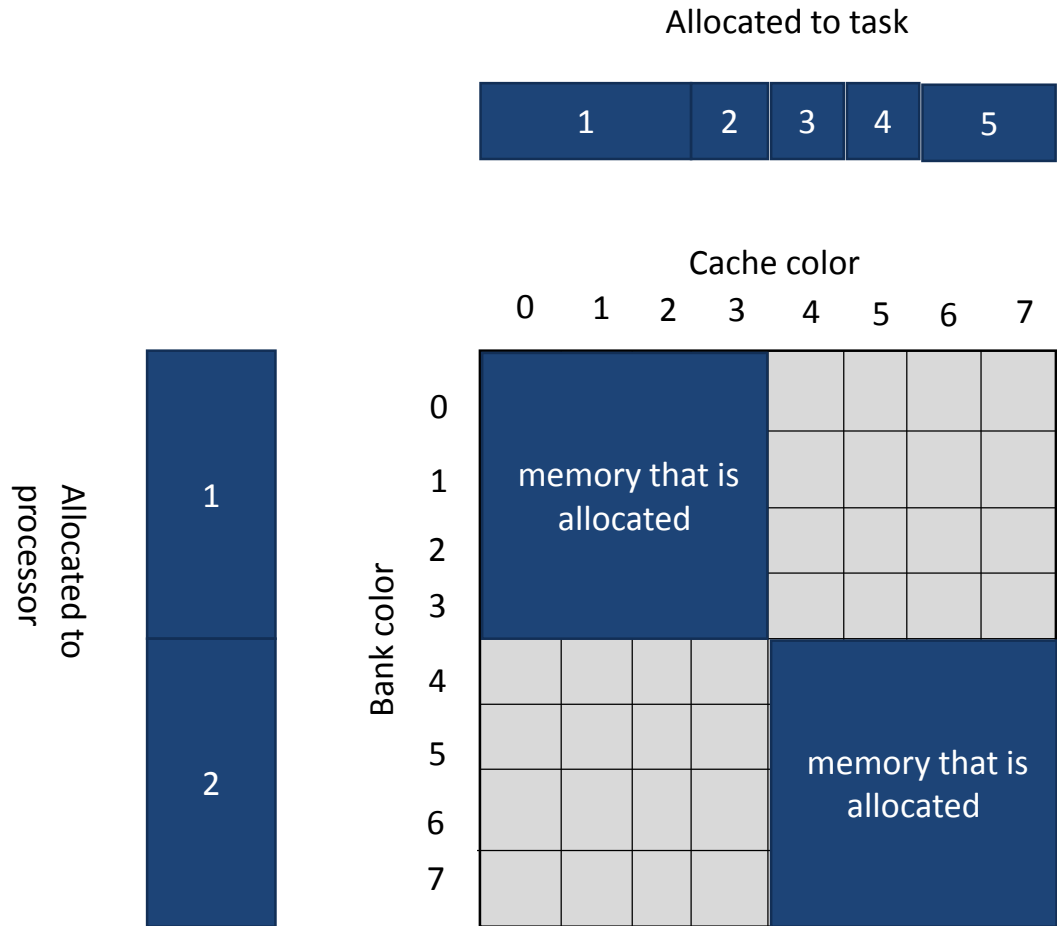
Assume task 1 and task 2 are assigned to processor 1 and task 3, task 4, and task 5 are assigned to processor 2.



Assume task 1 and task 2 are assigned to processor 1 and task 3, task 4, and task 5 are assigned to processor 2.



Assume task 1 and task 2 are assigned to processor 1 and task 3, task 4, and task 5 are assigned to processor 2.



Our coordinated cache and bank coloring forces us to waste 50% of the memory on a dual core system.

The configuration tool can be downloaded at

http://www.andrew.cmu.edu/user/banderss/software/coordinated_cache_and_bank_coloring_configuration/completeisolation.c.

Let us now see some screenshot from the use of the tool

```
ba@ba-desktop: ~/completeisolation$ ls -lrt
total 164
-rw-rw-r-- 1 ba ba 1048 Mar  5 16:31 taskset.txt
-rw-rw-r-- 1 ba ba 1048 Mar  7 21:00 taskset2.txt
dwxwxwx-x 5 ba ba 4096 Mar  7 22:19 ulimit00
-rw-r--r-- 1 ba ba 50891 Mar  7 22:35 completeisolation.c
-rwxrwx-x 1 ba ba 94880 Mar  7 22:36 completeisolation
ba@ba-desktop: ~/completeisolation$
```

```
Activities Terminal Mar 7 22:37 ba@ba-desktop: ~/completeisolation
1
4
32
16
B
1
B
1.000000 1.000000
B
B
B
0.300000 1
0.200000 2
0.100000 3
0.120000 4
0.080000 5
0.040000 6
0.030000 7
0.020000 8
2
0
1.990000 1.990000
B
B
1.900000 1
1.750000 2
1.740000 3
1.730000 4
1.720000 5
1.710000 6
1.700000 7
1.690000 8
3
7
1.000000 1.000000
B
B
0.250000 1
0.200000 2
0.160000 3
0.120000 4
0.110000 5
0.100000 6
0.090000 7
0.080000 8
4
5
1.990000 1.990000
B
B
1.850000 1
1.590000 2
1.580000 3
1.570000 4
1.560000 5
~More--(44%)
```

```
Activities Terminal Mar 7 22:37 ba@ba-desktop: ~/completeisolation
1.550000 0
1.540000 7
1.530000 8
5
4
100.000000 100.000000
B
B
30.000000 1
20.000000 2
16.000000 3
12.000000 4
11.000000 5
10.000000 6
9.000000 7
8.000000 8
6
2
199.000000 199.000000
B
B
B
190.000000 1
175.000000 2
174.000000 3
173.000000 4
172.000000 5
171.000000 6
170.000000 7
169.000000 8
7
3
100.000000 100.000000
B
B
30.000000 1
20.000000 2
16.000000 3
12.000000 4
11.000000 5
10.000000 6
9.000000 7
8.000000 8
B
B
1
199.000000 199.000000
B
B
185.000000 1
159.000000 2
158.000000 3
157.000000 4
156.000000 5
155.000000 6
154.000000 7
B~More--(98%)
```

```
Activities Terminal Mar 7 22:37 ba@ba-desktop: ~/completeisolation
1.540000 7
1.530000 8
5
4
100.000000 100.000000
B
B
30.000000 1
20.000000 2
10.000000 3
12.000000 4
11.000000 5
10.000000 6
9.000000 7
8.000000 8
6
2
199.000000 199.000000
B
190.000000 1
175.000000 2
174.000000 3
173.000000 4
172.000000 5
171.000000 6
170.000000 7
169.000000 8
7
3
100.000000 100.000000
B
B
30.000000 1
20.000000 2
16.000000 3
12.000000 4
11.000000 5
10.000000 6
9.000000 7
8.000000 8
B
1
199.000000 199.000000
B
B
185.000000 1
159.000000 2
158.000000 3
157.000000 4
156.000000 5
155.000000 6
154.000000 7
153.000000 8
ba@ba-desktop:~/completeisolation$
```

```
Activities Terminal Mar 7 22:37 ba@ba-desktop: ~/completeisolation
1.540000 7
1.530000 8
5
4
100.000000 100.000000
B
B
30.000000 1
20.000000 2
10.000000 3
12.000000 4
11.000000 5
10.000000 6
9.000000 7
8.000000 8
6
2
199.000000 199.000000
B
B
190.000000 1
175.000000 2
174.000000 3
173.000000 4
172.000000 5
171.000000 6
170.000000 7
169.000000 8
7
3
100.000000 100.000000
B
B
B
30.000000 1
20.000000 2
16.000000 3
12.000000 4
11.000000 5
10.000000 6
9.000000 7
8.000000 8
B
1
199.000000 199.000000
B
B
185.000000 1
159.000000 2
158.000000 3
157.000000 4
156.000000 5
155.000000 6
154.000000 7
153.000000 8
ba@ba-desktop:~/completeisolation$ ./completeisolation -i taskset.txt
```

```

157.000000 4
156.000000 5
155.000000 6
154.000000 7
153.000000 8
ba@ba-desktop: ~/completeisolation$ ./completeisolation -l taskset.txt
Starting tool that optimally assigns tasks and configures cache colors
Academic license - for non-commercial use only - expires 2021-05-04
Using license file /home/ba/gurobi.lic
Read LP format model from file ganilp0.lp
Reading time = 0.00 seconds
: 951 rows, 405 columns, 2865 nonzeros
Gurobi Optimizer version 9.1.1 build v9.1.1rc0 (linux64)
Thread count: 6 physical cores, 6 logical processors, using up to 6 threads
Optimize a model with 951 rows, 405 columns and 2865 nonzeros
Model fingerprint: 0xb0392555
Variable types: 45 continuous, 360 integer (352 binary)
Coefficient statistics:
  Matrix range    [5e-03, 2e+02]
  Objective range [8e+00, 8e+00]
  Bounds range    [1e+00, 1e+00]
  RHS range       [1e+00, 2e+02]
Presolve removed 85 rows and 5 columns
Presolve time: 0.01s
Presolved: 866 rows, 400 columns, 2780 nonzeros
Variable types: 0 continuous, 400 integer (352 binary)
Root relaxation: objective 0.000000e+00, 98 iterations, 0.00 seconds

  Nodes | Current Node | Objective Bounds | Work
 Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
    0     0   0.00000   0   34         -   0.00000   -   -   0s
    0     0   0.00000   0   25         -   0.00000   -   -   0s
H    0     0         0.0000000   0.00000   0.0000  0.00%   -   0s

Cutting planes:
  Gomory: 7
  Cover: 1
  Implied bound: 17
  Clique: 7
  MIR: 3
  Flow cover: 2
  Zero half: 7
  RLT: 49
  Relax-and-lift: 3

Explored 1 nodes (296 simplex iterations) in 0.05 seconds
Thread count was 6 (of 6 available processors)

Solution count 1: 0

Optimal solution found (tolerance 1.00e-04)
Best objective 0.000000000000e+00, best bound 0.000000000000e+00, gap 0.00000
ba@ba-desktop: ~/completeisolation$

```

```
Activities Terminal Mar 7 22:37 ba@ba-desktop: ~/completeisolation
Flow cover: 1
Zero half: 7
RLT: 48
Relax-and-lift: 3

Explored 1 nodes (290 simplex iterations) in 0.05 seconds
Thread count was 6 (of 6 available processors)

Solution count 1: 0

Optimal solution found (tolerance 1.00e-04)
Best objective 0.00000000000e+00, best bound 0.00000000000e+00, gap 0.0000%
ba@ba-desktop: ~/completeisolation$ more taskset_results_from_analysis.txt
F. Feasible assignment found.
Task with id=1, priority = 8, T=1.000000, D=1.000000, ncells = 8
R = -1.000000
selectedexecutiontime = 0.020000
processor_assigned_to = 1
selectednumberofcachepartitions = 8
Task with id=2, priority = 6, T=1.990000, D=1.990000, ncells = 8
R = -1.000000
selectedexecutiontime = 1.730000
processor_assigned_to = 1
selectednumberofcachepartitions = 4
Task with id=3, priority = 7, T=1.000000, D=1.000000, ncells = 8
R = -1.000000
selectedexecutiontime = 0.200000
processor_assigned_to = 2
selectednumberofcachepartitions = 2
Task with id=4, priority = 5, T=1.990000, D=1.990000, ncells = 8
R = -1.000000
selectedexecutiontime = 1.590000
processor_assigned_to = 3
selectednumberofcachepartitions = 2
Task with id=5, priority = 4, T=100.000000, D=100.000000, ncells = 8
R = -1.000000
selectedexecutiontime = 11.000000
processor_assigned_to = 4
selectednumberofcachepartitions = 5
Task with id=6, priority = 2, T=199.000000, D=199.000000, ncells = 8
R = -1.000000
selectedexecutiontime = 171.000000
processor_assigned_to = 4
selectednumberofcachepartitions = 6
Task with id=7, priority = 3, T=100.000000, D=100.000000, ncells = 8
R = -1.000000
selectedexecutiontime = 20.000000
processor_assigned_to = 3
selectednumberofcachepartitions = 2
Task with id=8, priority = 1, T=199.000000, D=199.000000, ncells = 8
R = -1.000000
selectedexecutiontime = 159.000000
processor_assigned_to = 2
selectednumberofcachepartitions = 2
ba@ba-desktop: ~/completeisolation$
```

Potential use in a DevOps pipeline

1. In each thread/process, the source code needs to be modified to that before the thread/process starts execution, it reads from a text file to get information. With this information, it calls RK-functions to setup the coordinate cache and bank coloring and also assign it to processor core. Each thread/process has its own file that it reads from in order to get this info to pass to RK-functions. Each thread/process also has an associated file that describes parameters (e.g., period, execution time with different cache colors).
2. When “make” is run, a configuration preparation tool (not yet developed) is invoked. It gathers information about threads/processes from the aforementioned files and puts this info to a single file. This single file is input to the configuration tool (which we already have). Then the configuration tool generates the configuration to a file. A configuration effector tool (not yet developed) reads this file and writes to various files for each thread/process.