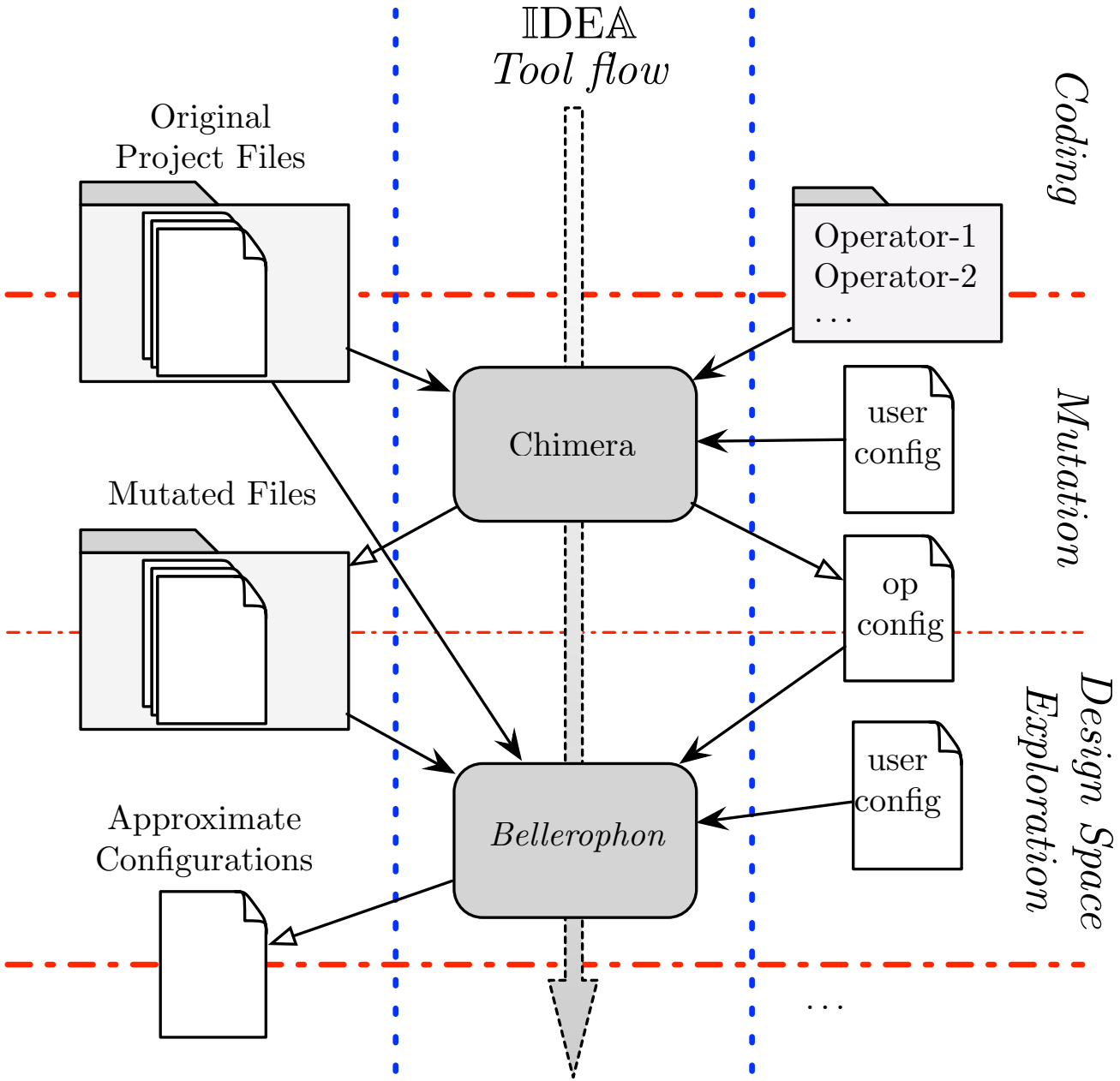
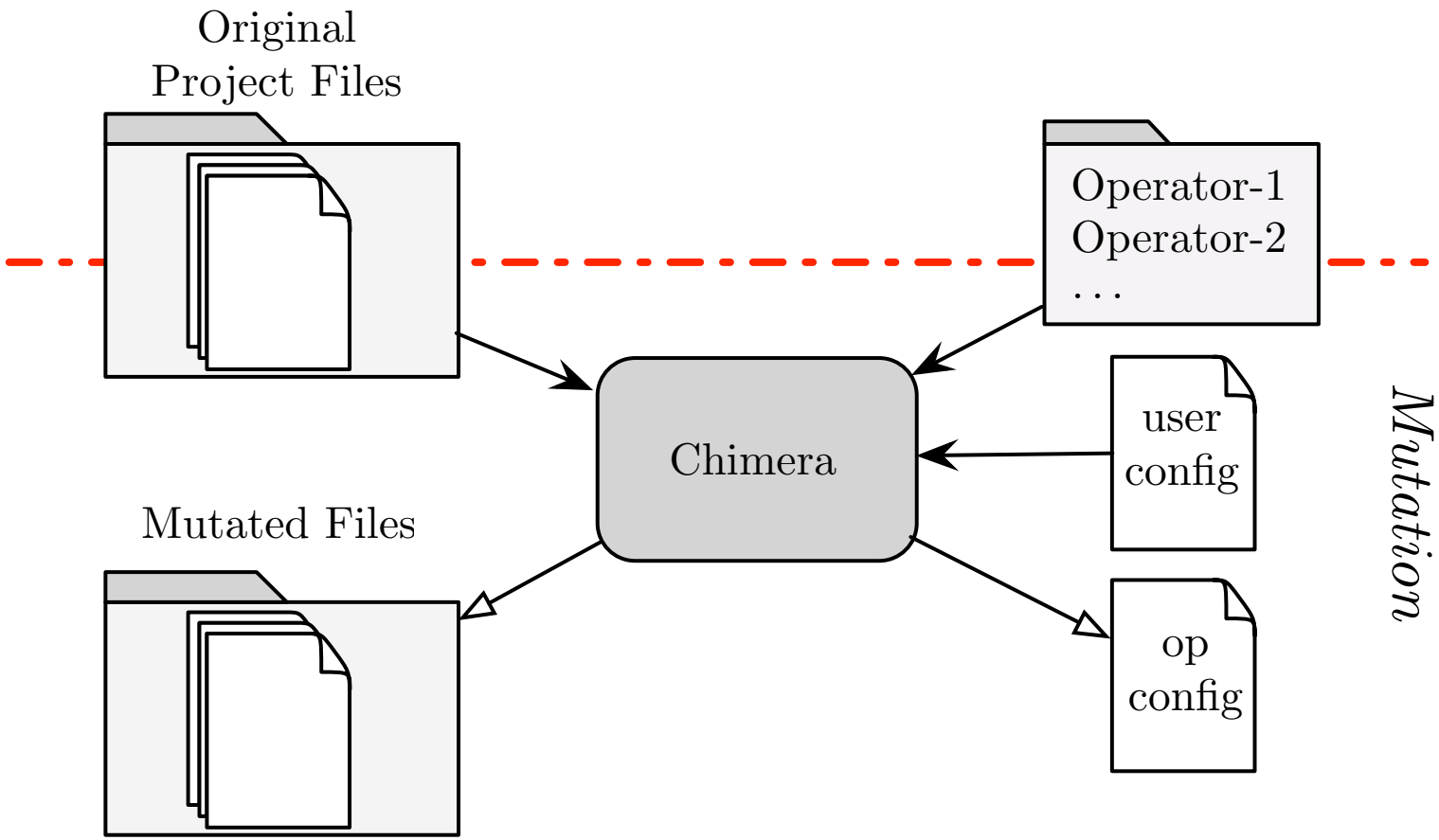
IIDEAA

- **IIDEAA: Is a Design Exploration tool for Approximating Algorithms**
- **It is composed of two tools:**
  - **Chimera**: mutation engine;
  - **Bellerophon**: search engine;

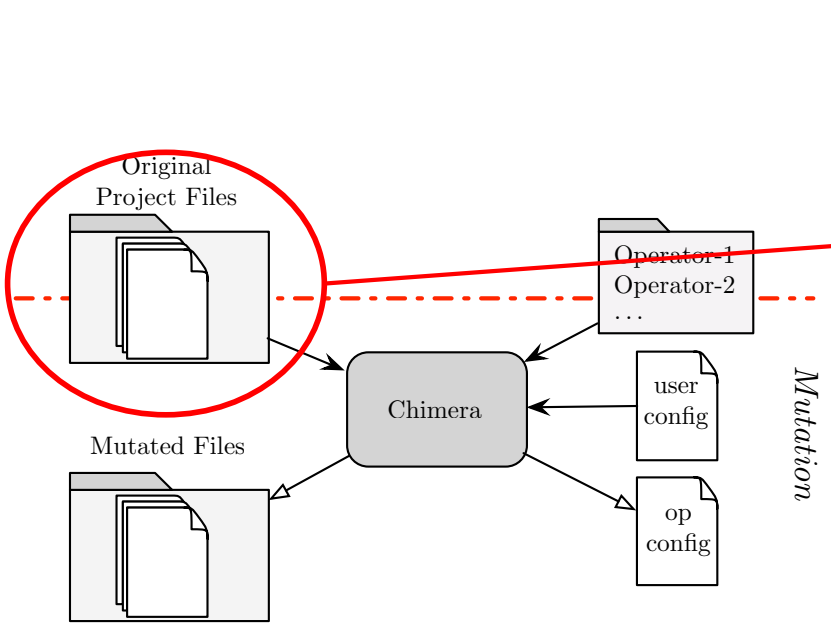
IIDEAA Flow

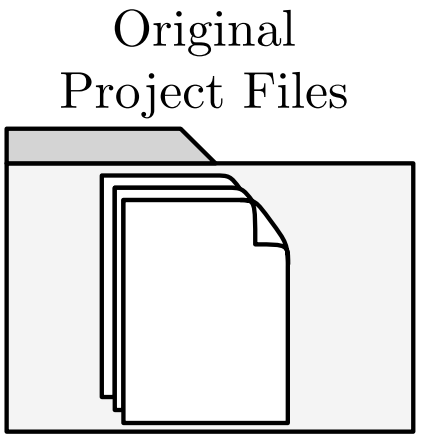

IDEA
Tool flow
Coding
Mutation
Design Space Exploration
Original Project Files
Operator-1
Operator-2
...
Chimera
user config
Mutated Files
op config
user config
Bellerophon
Approximate Configurations
...

1) Mutation


Original
Project Files
Operator-1
Operator-2
...
Chimera
user
config
op
config
Mutated Files
Mutation

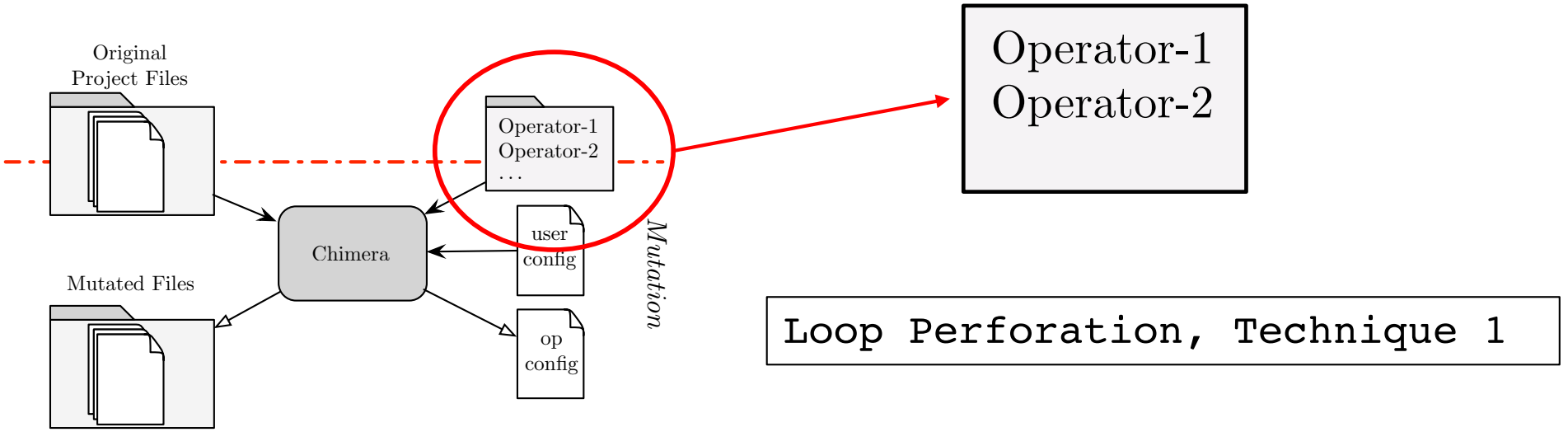
Example: Inputs



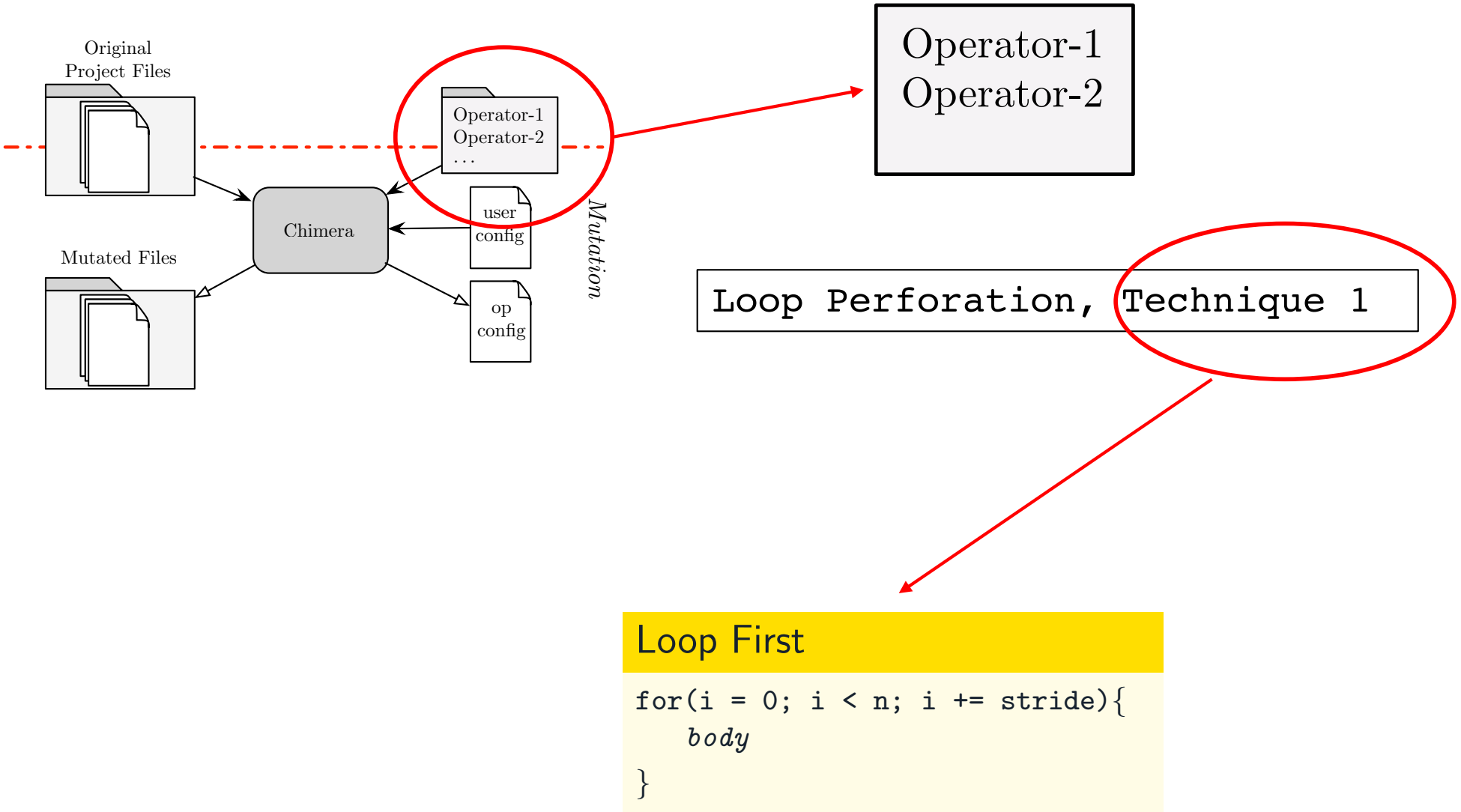



```
void f1 () {
..
..
for (i = 0; i < N; i++) {
        …
}

main () {
…
f1 ();
..
}
```

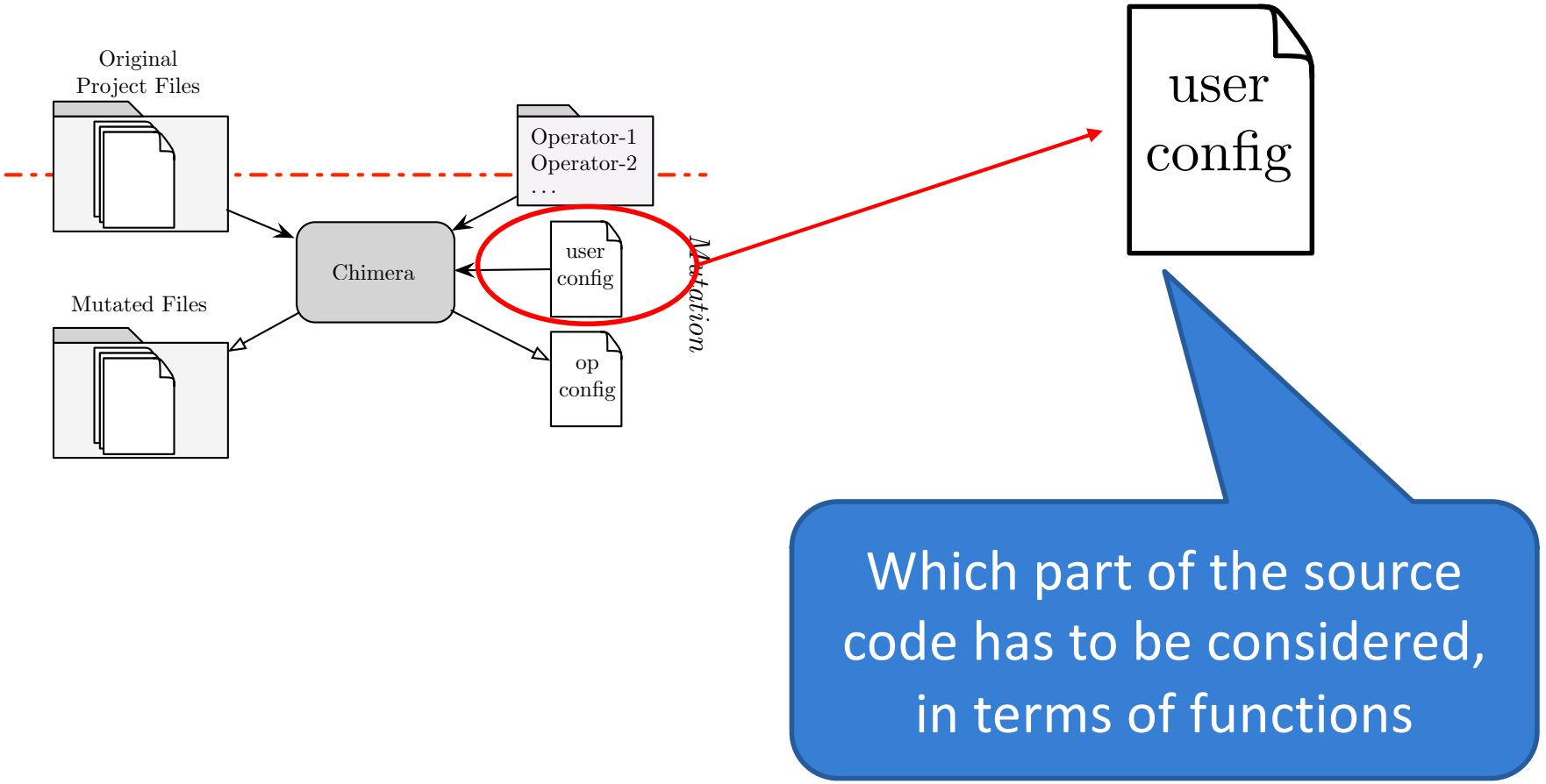
Example: Inputs



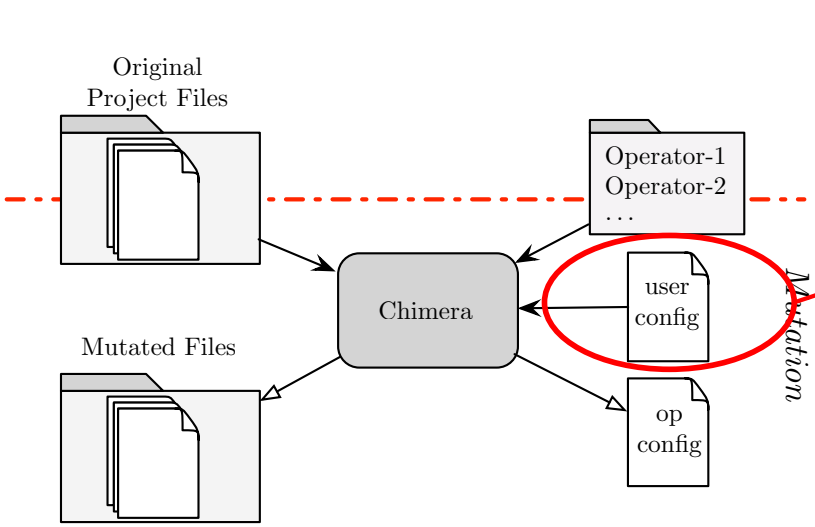
Example: Inputs



1) Mutation: Example



Example: Inputs

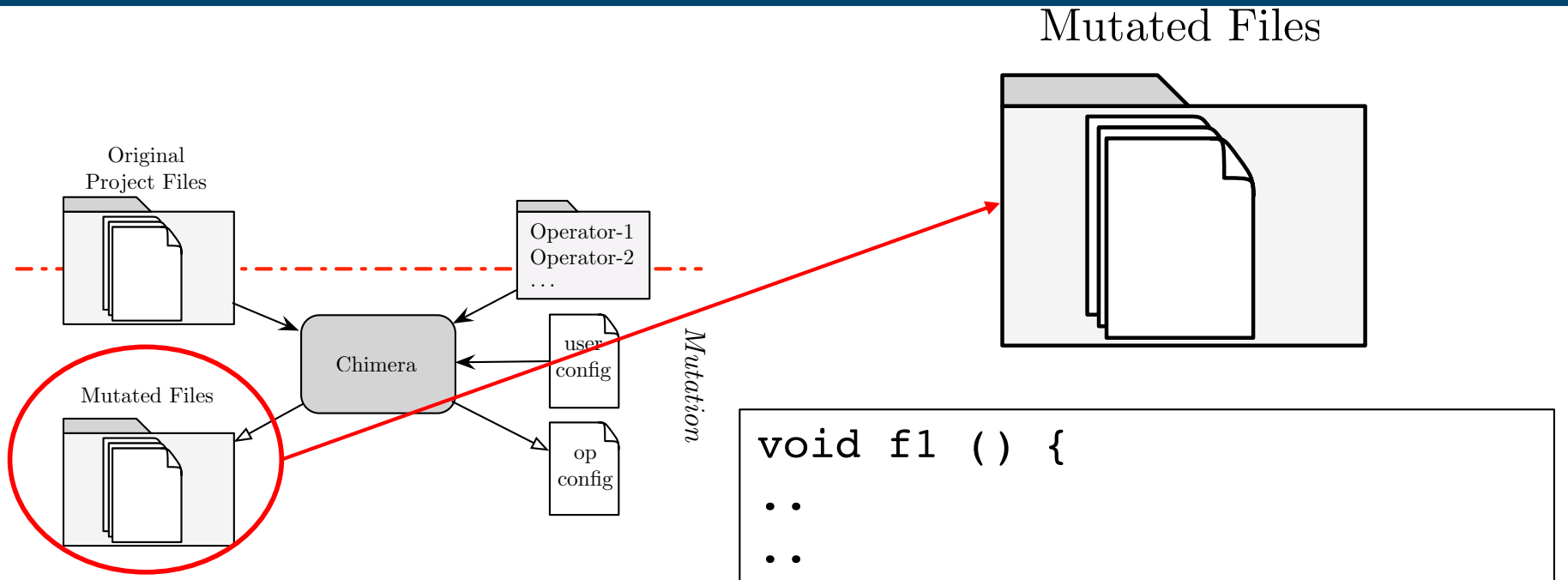


user
config

f1

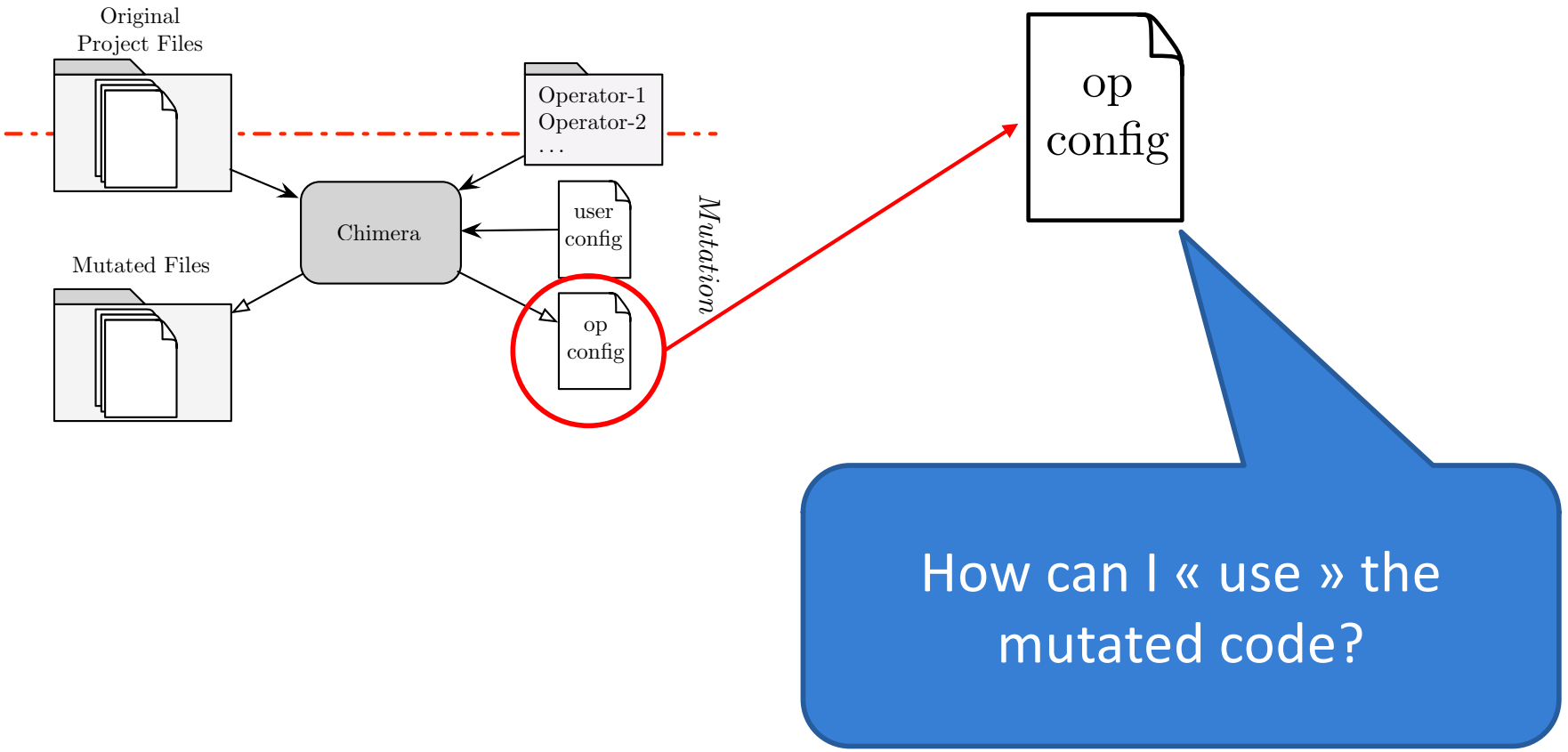
```
void f1 () {
}

main () {
}
```

Example: Outputs

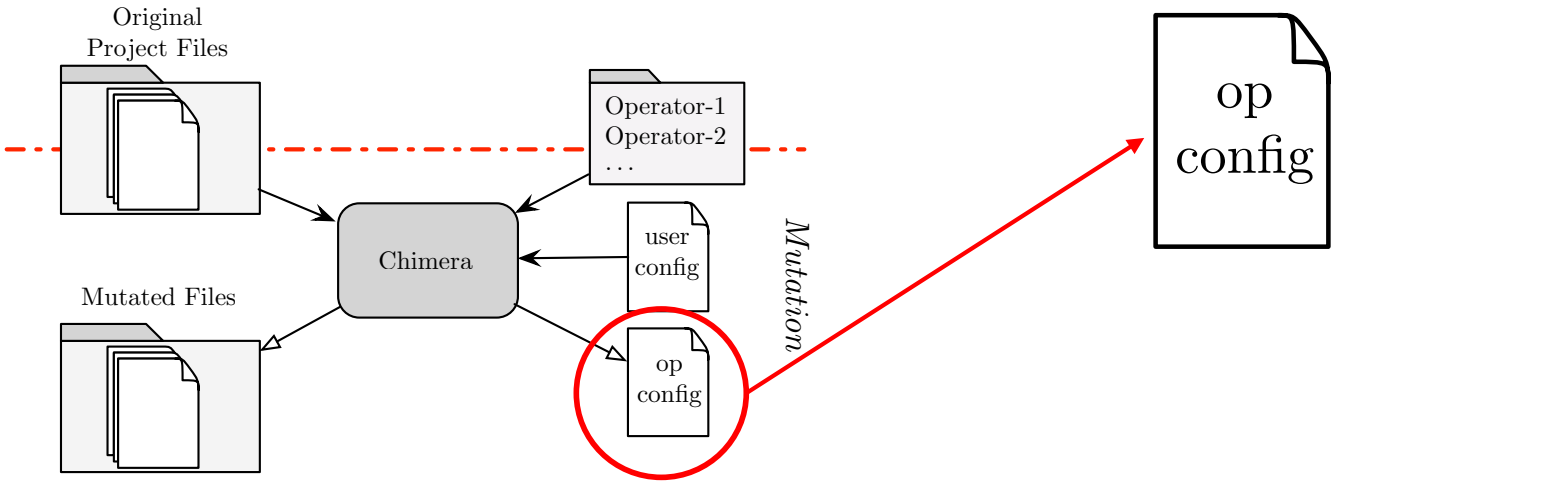



```
void f1 () {
..
..
for (i = 0; i < N; i+=stride)
{
        …
}

main () {
…
f1 ();
..
}
```

Example: Outputs



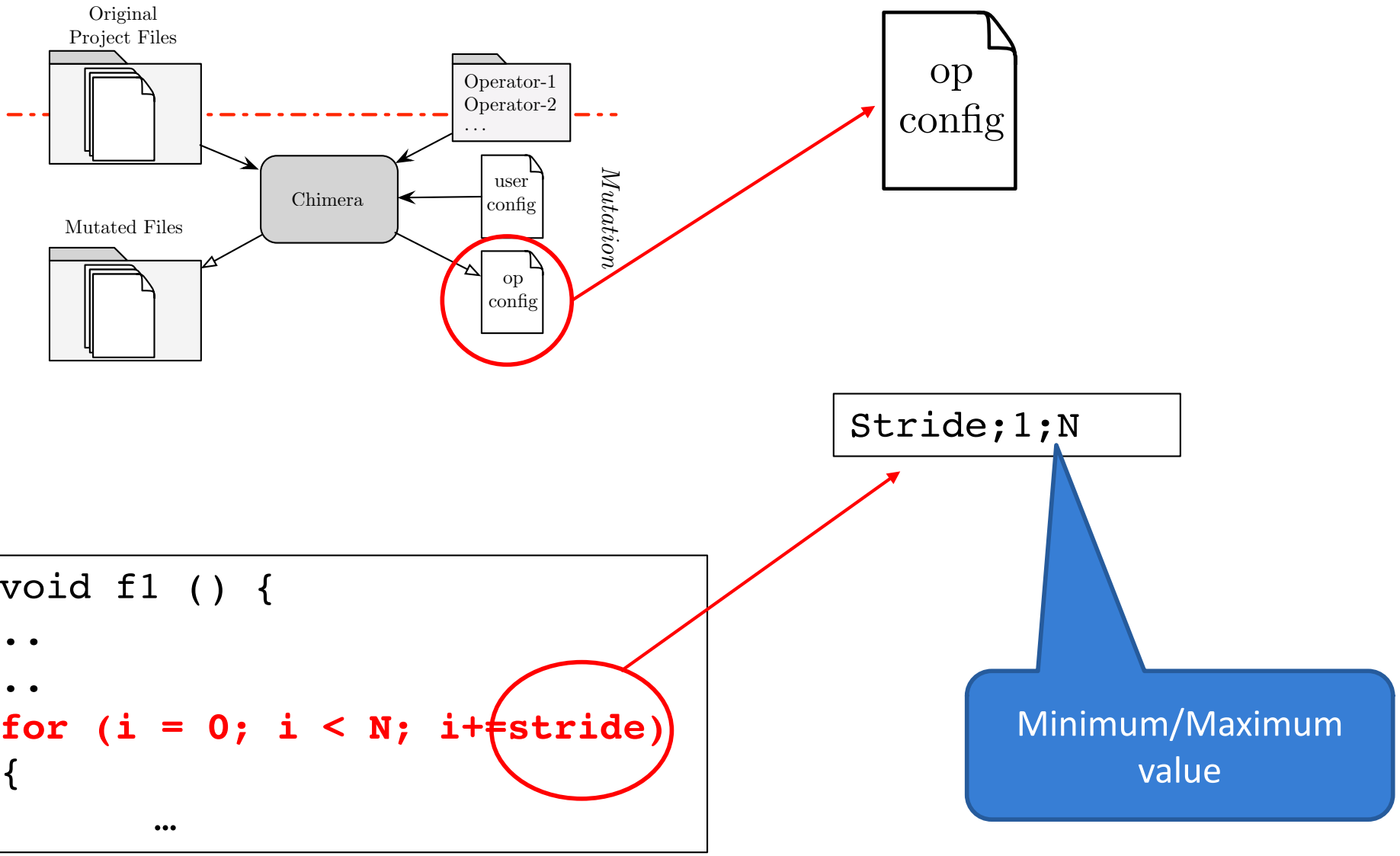
Example: Outputs



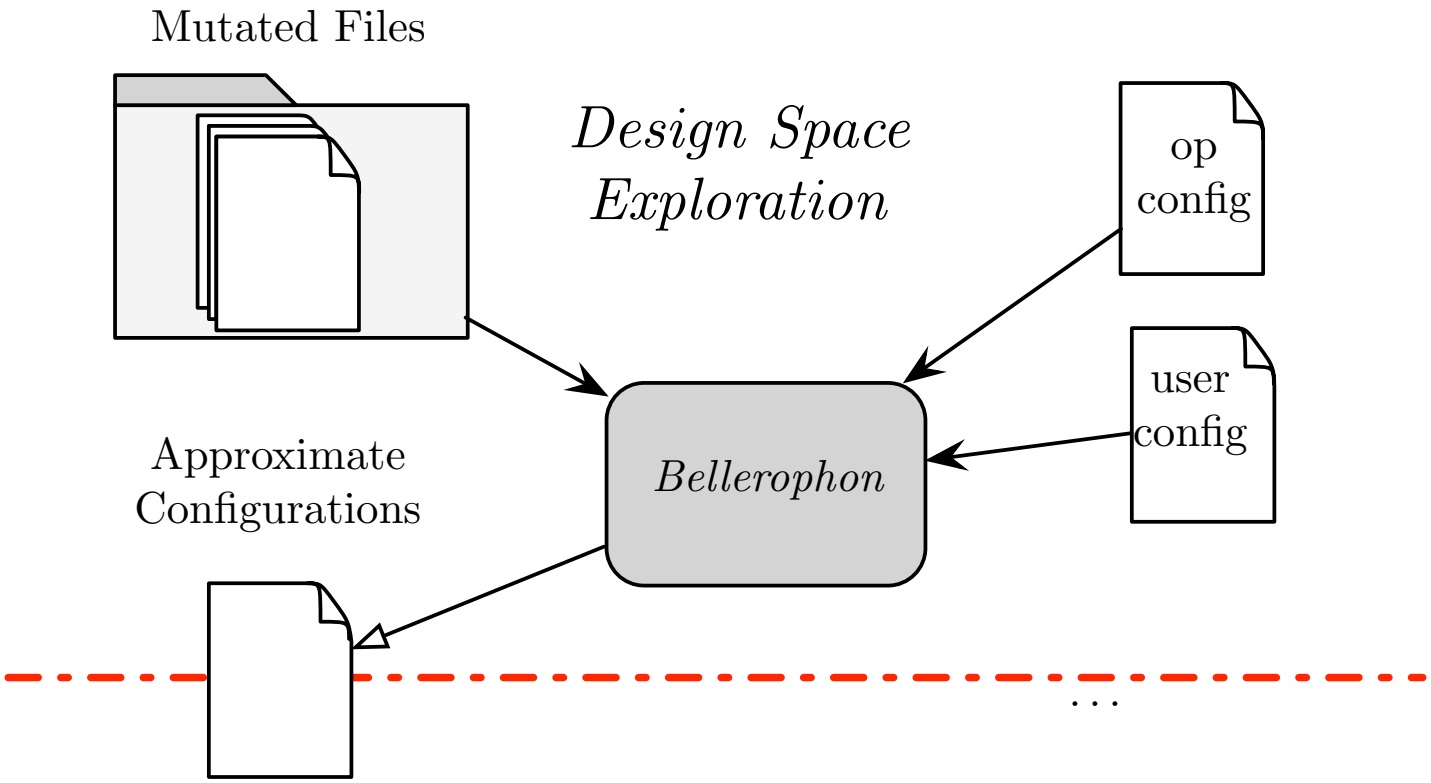

```
Stride;1;N
```

```
void f1 () {
..
..
for (i = 0; i < N; i+=stride)
{
        …
```

Example: Outputs

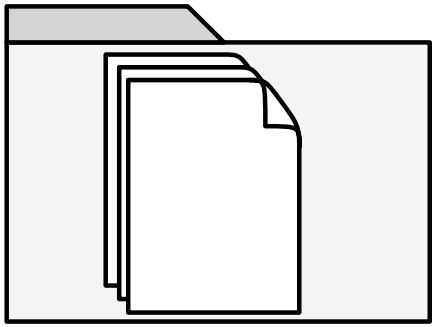


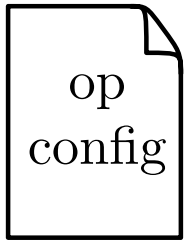
2) Search


Mutated Files
Design Space Exploration
op config
user config
Bellerophon
Approximate Configurations
...

Example: Inputs

Mutated Files

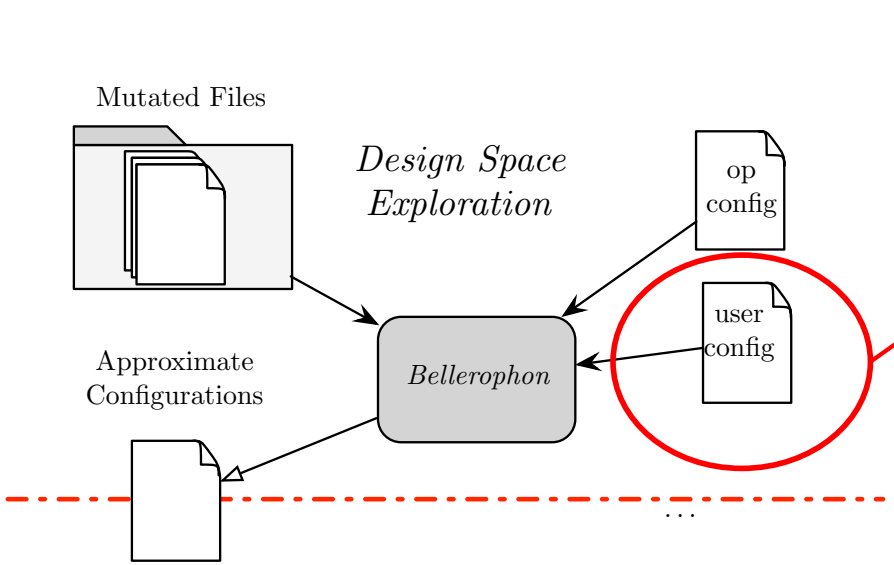


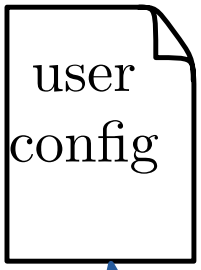



```
void f1 () {
..
..
for (i = 0; i < N; i+=stride)
{
        …
}

main () {
…
f1 ();
..
}
```

```
Stride;1;N
```

Example: Inputs





Evalute a given **solution** in terms of:

- Perfomances
- Accuracy

What is a solution?

- **A solution is an assignement to the used AxC operators**
- **Example**
  - **The operator is the loo perforation via the *stride* variable**

```
Stride;1;N
```

```
for (i = 0; i < N; i+=stride)
```

  - **Stride = 1**; is a solution corresponding to the precise code
  - **Stride = 2;** is a solution where 50% of iterations are executed
  - **Stride = 3;** is a solution where N/3 of iterations are executed
  - **…**
  - **Stride = N;** is a solution where 0 iterations are executed

What is a solution?

- Each Solution corresponds to an Approximation of the input algorithm.
- The Search Engine has to select the « best » one
  - Maximize performances
  - Minimize the InAccuracy

What is a solution?

- Each Solution corresponds to an Approximation of the input algorithm.
- The Search Engine has to select the « best » one
  - Maximize performances
  - Minimize the Inaccuracy

Need for quantifying the performances and the accuracy

Performances

- Let’s coming back to our example…

```
for (i = 0; i < N; i+=stride)
```

  - Higher the stride’ value higher the performances
    - Less iterations
- …Maximize performances = Maximize *stride*

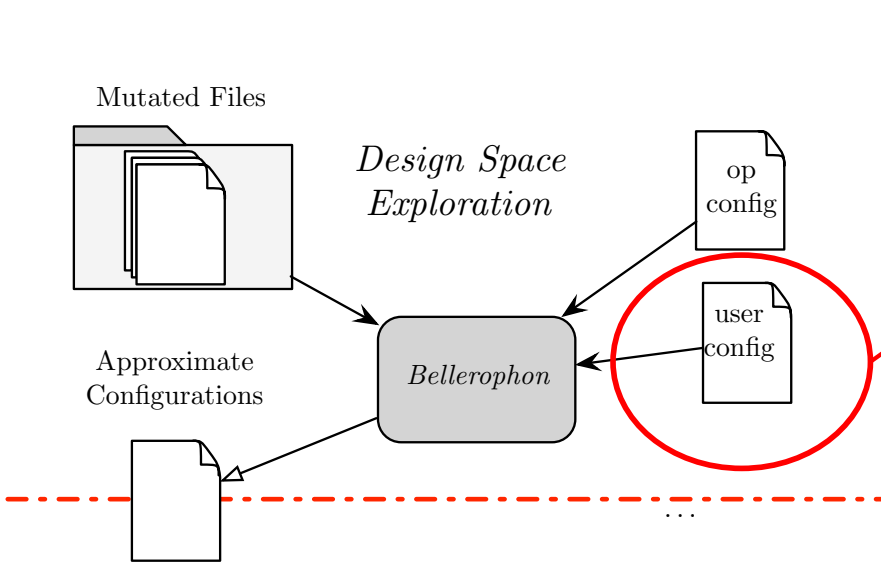
Accuracy

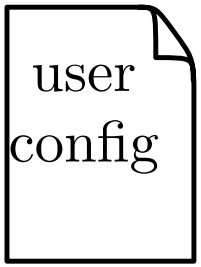
- Let's coming back to our example…

```
for (i = 0; i < N; i+=stride)
```

- What about the impact of loop perforation on the accuracy?
  - **It depends on the computation done in the loop**
- **The user has to specify how compute the inaccuracy**

Example: Inputs

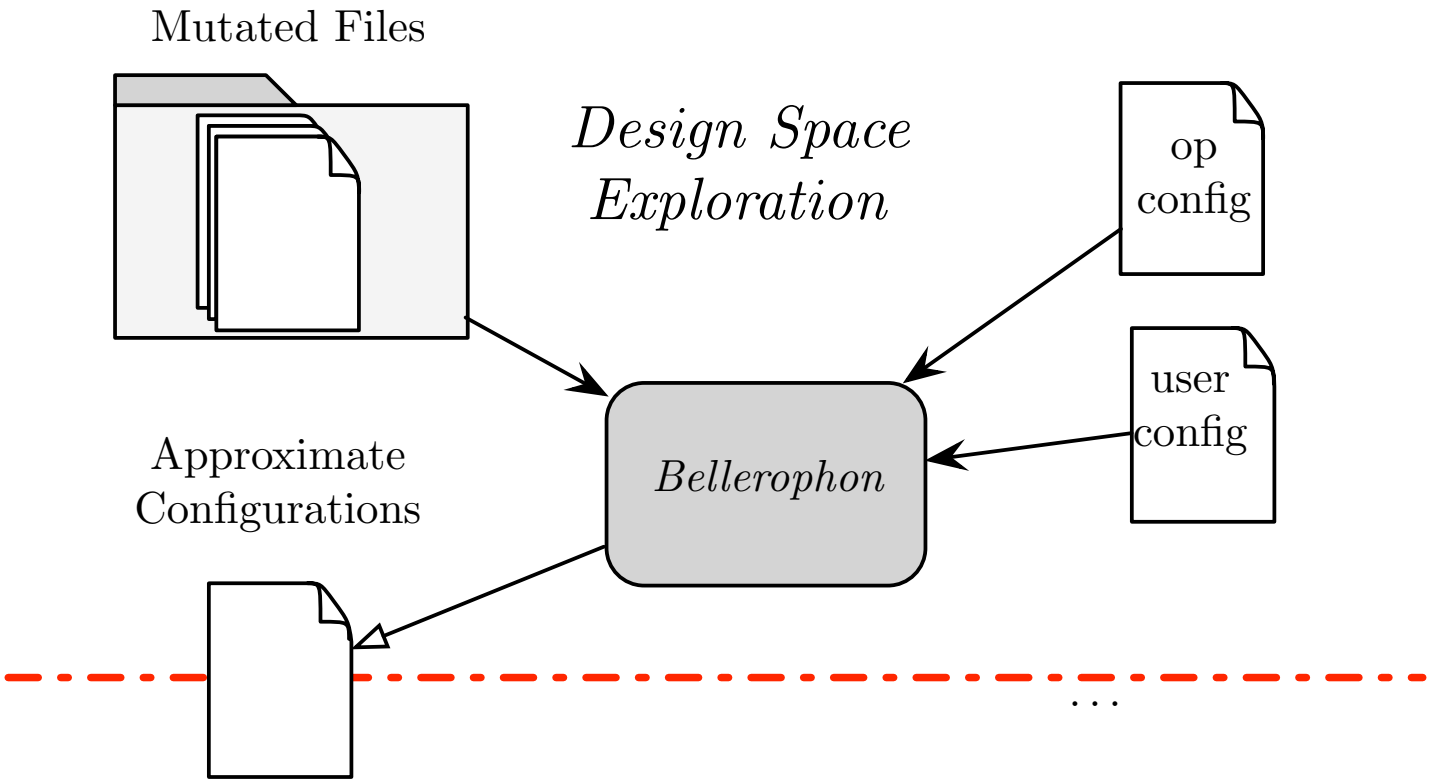




Perfomances (Reward): Maximize stride

Acciracy(Error): <user defined>

2) Search


Mutated Files
Design Space Exploration
op config
user config
Bellerophon
Approximate Configurations
...

2) Search

- **Exploit ParadisEO**
- **Modelling:**
  - **Chromosomes:** each individual is modeled as an integer array. The size is the number of mutators;
  - **Population:** number of mutated versions. Each individual is characterized by its set of chromosomes;
  - **Variation Operations:** the crossover and bit-flip mutation have been chosen.

2) Search

- **Exploit ParadisEO**
- **Modelling:**
  - **Chromosomes:** ea… an integer array. The si… is t… of mutates)
  - **Population:** number of mutated versions. Each individual is

Example of Population

Individual	Stride1	Stride2	Reward
0	3	5	8
1	2	9	11
⋮	⋮		⋮
k-1	5	2	7

…tion

DEMO (Linux)

- **Install the tool**

```
git clone https://github.com/mariobarbareschi/iideaa-docker
```

- **Install the docker container**

```
docker pull mariobarbareschi/iideaa
```

DEMO (Linux)

- **Run the container**

docker run -dit -v <local_application_demo>:/opt mariobarbareschi/iideaa

- **Attach the container**

docker ps #get the running **docker id**

docker attach <**docker id**>

DEMO

- **Application: Newton-Raphson Method**

DEMO

- **Application demo:**
- **3 directories**
  - **src**
  - **clang-chimera**
  - **bellerophon**

DEMO

- **Application demo:**
- **3 directories**
  - **src**
  - **clang-chimera**
  - **bellerophon**

DEMO

- **Source Code**

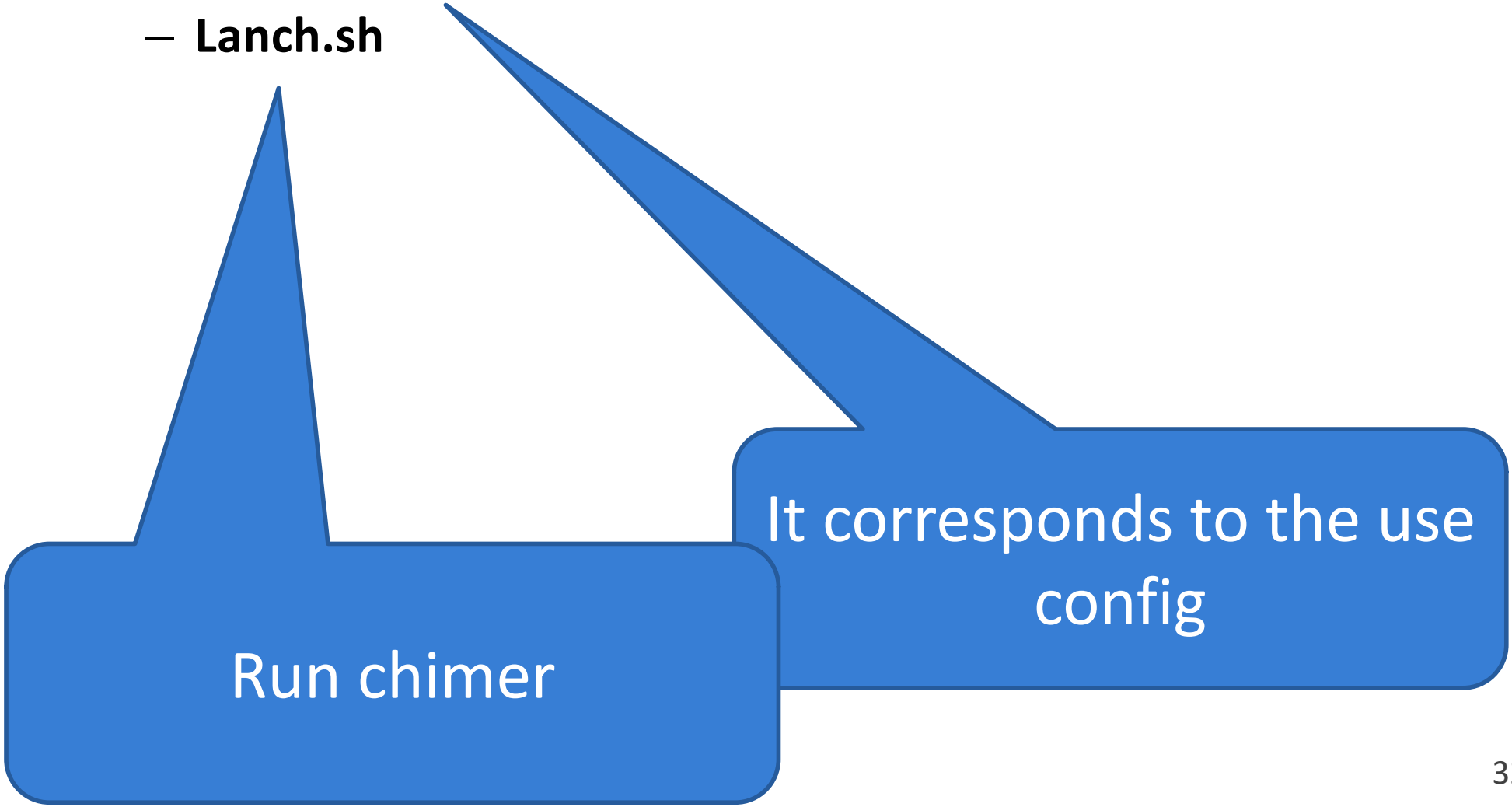
```c
#include<stdio.h>
#include<math.h>

float f(float x);
float df (float x);

float  newton()
{
    int itr, maxmitr;
    float h, x0, x1, allerr;


    maxmitr = 120;
    x0 = 45.4;
    for (itr=1; itr<=maxmitr; itr++)
    {
        h=f(x0)/df(x0);
        x1=x0-h;
        x0=x1;
    }
    return x0;
}
```

DEMO

- **Application demo:**
- **3 directories**
  - **src**
  - **clang-chimera**
  - **bellerophon**

DEMO

- **Clang-chimera:**
  - **Conf.csv: newton,LoopPerforationOperator1**
  - **Lanch.sh**



DEMO

- **Chimera Output**
  - **loop_report.csv: stride1,19,U,-9999**
  - **newton.cpp**

```
12 float  newton()
13 {
14     int itr, maxmitr;
15     float h, x0, x1, allerr;
16
17
18     maxmitr = 120;
19     x0 = 45.4;
20     for (itr=1; itr<=maxmitr; itr = itr + stride1)
21     {
22         h=f(x0)/df(x0);
23         x1=x0-h;
24         x0=x1;
25     }
26     return x0;
27 }
```

DEMO

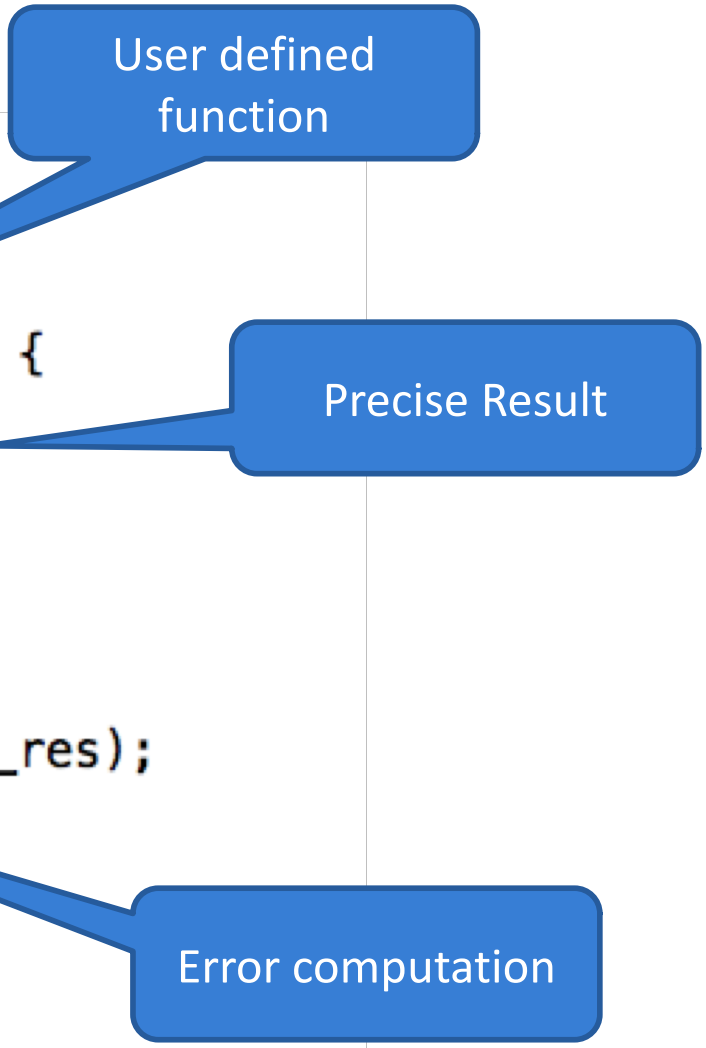
- **Bellerophon**
  - **bellerophon_functions.cpp**

```
 5
 6 float  newton ();
 7 using namespace std;
 8
 9 extern "C" double BELLERO_getError() {
10
11         float golden_res = 2.740646;
12         float axc_res;
13         float error;
14
15         axc_result = newton ();
16         error = abs(axc_res - golden_res);
17
18         return (error);
19
20 }
21
```

DEMO

- **Bellerophon**
  - **bellerophon_functions.cpp**

```
 5
 6 float  newton ();
 7 using namespace std;
 8
 9 extern "C" double BELLERO_getError() {
10
11         float golden_res = 2.740646;
12         float axc_res;
13         float error;
14
15         axc_result = newton ();
16         error = abs(axc_res - golden_res);
17
18         return (error);
19
20 }
21
```



DEMO

- **Bellerophon**
  - **gaconfig.param:** Paradiseo parameters

```
######                  General    ######
--help=0                                    # -h : Prints this message
--stopOnUnknownParam=1                      # Stop if unkown param entered
--seed=1203083493                         # -S : Random number seed

######    Evolution Engine          ######

--popSize=500                               # -P : Population Size

######    Stopping criterion        ######
--maxGen=10                                # -G : Maximum number of generations

######    Variation Operators       ######
--crossRate=1                               # Relative rate for the only crossover
--shiftMutRate=0.5                          # Relative rate for shift mutation
--exchangeMutRate=0.5                       # Relative rate for exchange mutation
--pCross=0.25                               # -c : Probability of Crossover
--pMut=0.35                                 # -m : Probability of Mutation
```

DEMO

- **Bellerophon**
  - **Launch.sh**: run the tool

```
[BELLEROPHON INFO] Acceptable solutions:
```

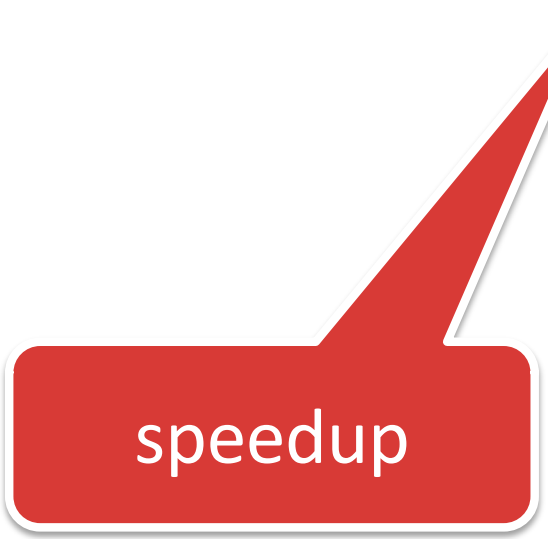
Error	Reward	Penalty	#OP	OP_0
2.38419e-07	9	0	1	9

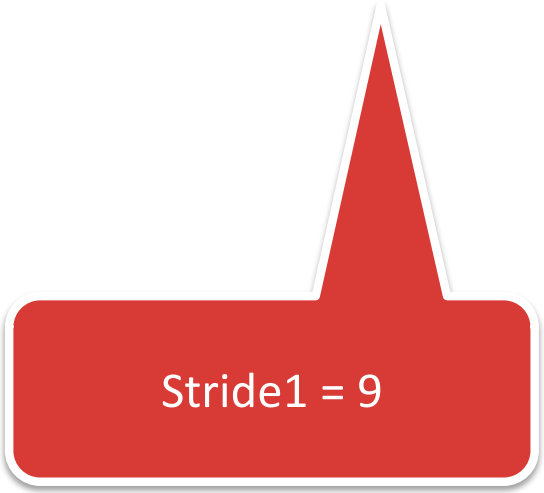
DEMO

- **Bellerophon**
  - **Launch.sh**: run the tool

```
[BELLEROPHON INFO] Acceptable solutions:
```

Error	Reward	Penalty	#OP	OP_0
2.38419e-07	9	0	1	9





DEMO

- **Application K-means (custom floating point)**
- **3 directories (same flow as for the first newton)**
  - **Code**
  - **Chimera**
  - **bellerophon**
- **3 « extra directories »**
  - **execution**
  - **Scripts: pyhton script for converting**
  - **Images: workload image**

DEMO

- **After executing kmeans the output will be somthing similar**
  - **[BELLEROPHON INFO] Acceptable solutions:**
  - **Error Reward Penalty #OP OP_0 OP_1 OP_2 OP_3 OP_4 OP_5**
  - **0.0077334 936 0 6 13 19 20 20 22 0**
  - **0.0114403 953 0 6 4 0 3 21 19 12**
  - **0.0174854 977 0 6 13 19 20 21 22 0**
  - **0.00367057 785 0 6 16 19 20 17 21 0**
  - **0.10167 1096 0 6 4 3 20 22 12 21**
  - **0.00415019 788 0 6 16 19 20 16 21 6**
  - **0.00331577 784 0 6 16 18 20 17 21 0**
  - **0.0322582 1029 0 6 14 1 10 22 6 22**
  - **0.028161 1022 0 6 1 13 18 18 21 15**
  - **0.0520083 1094 0 6 4 0 21 22 12 21**
  - **0.000343428 752 0 6 21 19 22 11 20 13**
  - **0.124375 1098 0 6 4 4 21 22 12 21**

- **Each line corresponds to a solution:**
  - **One solution is composed of six values corresponding to the OP_0 OP_1 OP_2 OP_3 OP_4 OP_5 custom float variables**

DEMO

- In the execution directory, you have to copy and paste the solutions into the operators_values.csv file
- The scripts create_variants.sh has to be executed to create n versions of the kmeans application, one per each solution. The script also runs the application
- The n versions with input/output rgb image will be stored in the variants directory
- Use the convert2png.sh to convert each variant's output into a png file
- Navigate the subdirectories and check the results

Kmeans output examples

- We ran the IDEA flow over a k-means algorithm:
  - **Quality function**: MSE over 20 different images;
  - **Threshold**: 0.1;
  - **Mutators**: bit-width reduction;
  - **Reward function**: **additions/subtractions**: number of bits saved **multiplications/divisions**: squared number of bits saved.

Try yourself….