Greedy

To read more about Clustering please check our paper at https://www.usenix.org/conference/fast22/presentation/kisous "The what, The from, and The to: The Migration Games in Deduplicated Systems". In 20th USENIX Conference on File and Storage Technologies (FAST 22), 2022

Authors:

Roei Kisous, Ariel Kolikant, Gala Yadgar (Technion - Israel Institute of Technology);

Abhinav Duggal (Dell Technologies);

Sarai Sheinvald (ORT Braude College of Engineering).

Guide Author: Ariel Kolikant

1) Introduction:

In this guide, we go over the requirements, installations, and phases needed for the greedy algorithm to run on the experiments presented in the article.

The greedy algorithm works in iterations, in which the "best" file migration is selected for each iteration. Iteration type influences the method for calculating the best migration. During balancing iterations, the best migration will be the one that brings the system closer to balance; during optimization phases, the best migration will be the one that reduces the total system size without breaking the load balance constraints.

The flow of running the Greedy algorithm is

1) Build the greedy algorithm code by:
    a. sudo apt-get install libboost-dev
    b. running the Makefile
2) Create volumes compatible with the used standard. Explained in the volume standard section.
3) Create a volume list input file. Explained in the volume list section
4) Run greedy, with the volume list as input. Explained in the running greedy section. The output is explained in the output section.
5) Run calc with the migration plan as input. Explained in the running calc section.

We will use example files for a walkthrough example in the walkthrough section.

2) Files you will need:

1) GreedyLoadBalancerUnited.cpp
2) Makefile
3) Calc

3) walkthrough

1) **Build the greedy algorithm code by running the Makefile:**

   run command `make`
   This will create the file GreedyLoadBalancerUnited

2) **Create volumes compatible with the used standard. Explained in the volume standard section:**

   the example files are called basic1.csv, basic2.csv, basic3.csv

3) **Create a volume list input file. Explained in the volume list section:**

   the example file is called basicLoadBalance_k1_3Vols

4) **Run greedy, with the volume list as input. Explained in the running greedy section. The output is explained in the output section.**

   run command `./GreedyLoadBalancerUnited basicLoadBalance_k1_3Vols out.csv summary.csv 21600 20 0.25`

This will create the files summary.csv, out.csv, out.csv_migrationPlan

5) **Run calc with the migration plan as input. Explained in the running calc section.**

Run command `calc -file out.csv_migrationPlan -output calcResult`
This will create the file calcResult

4) Volume Standard:

The volumes file represents the initial state of a volume within the system. It tells us what files are in the volume and what blocks those files contain. Objects have unique ID's shares across volumes, but do not necessarily need to have the same SN across multiple volumes.

1) Header:

```
#Output type: block-level
#Input files: 0002
#Target depth: 10
#Num files: 36
#Num directories: 29
#Num Blocks: 268
```

2) File lines:

F (Object type), 0 (SN), 002_281474976710693 (File ID), 0 (Parent Directory SN), 11 (Num of base objects), 0 (Base Object SN), 16046 (Base Object size), 1 (Base Object SN), 4096 (Base Object size), ...

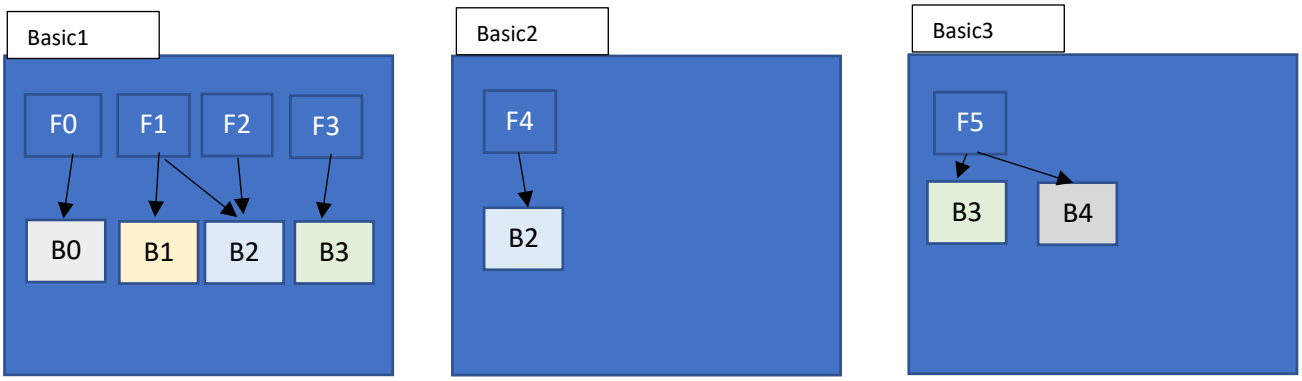
3) Block lines:

B/P (Object type), 77 (SN), 9029760f86 (Object ID), 5 (Shared by num files), 26 (file SN), 31 (file SN), 57 (file SN), ...

4) Folders:

D/R (Object type), 2 (SN), 002_281474976710694 (Directory ID), 0 (Parent Directory SN), 2 (Num of Subdirs), 4 (Num of Files), 3 (Dir SN), 5 (Dir SN), 17 (File SN), 21 (File Sn), ...

An example can be found in examples/basic1.csv, examples/basic2.csv, examples/basic3.csv.



5) Volume List:

The purpose of the algorithm is the creation of a migration plan which would lead to the system reaching a certain load balance. The layout of that desired balance is given in the file called volume list and is given as input to the algorithm.

!!! each column must be separated with ", " which is a comma and a space!!!

```
/basic_1.csv, a, 0.20
/basic_2.csv, a, 0.20
/basic_3.csv, a, 0.60
```

The first column is the path to the files containing the volume in the volume standard, explained in the volume standard section.

The second column is the role the volume plays in the algorithm. Roles can be one of: a, s, t

s: source - can only have files moved **from** itself.
t: target - can only have files moved **into** itself.
a: all – can have files moved both from and into itself.

For reconstructing the experiments, only a is ever relevant.

The third column is the desired fraction the volume should have after the migration plan is enacted. These need to be between 0 and 1 and have a total of 1.

For the sake of creating summary files; explained in the summary file section; there exists a naming standard on the volume list files: [name]_k[sampleSize]_[volume number]Vols

An example is given in examples/basicLoadBalance_k1_3Vols.

6) Running Greedy:

A couple of things to take in mind before starting the runs.

1) Summary file: a **csv** file where the statistics for **all** your runs will be stored. explained in the summary file section
2) Output folder: where you want the migration plans to be saved. The files are explained in the output section.

To run: ./GreedyLoadBalancerUnited {volumelist} {output} {summaryFile} {timelimit} {Traffic} {margin}

For example ./GreedyLoadBalancerUnited basicLoadBalance_k1_3Vols outputFile.csv summaryFile.csv 21600 20 0.3

1) Volumelist: volume List section
2) Output: the path to the where the csv migration plan is to be written.
3) summaryFile: path to the summary file.
4) TimeLimit: after how many seconds should the algorithm stop
5) Traffic: what is the maximum amount of traffic allowed for the algorithm (0-100, representing percentage)
6) Margin: the error margin allowed from the desired balance. If this margin is 1 or more, the code would run without load balancing. (0-1, representing fraction)

7) Summary File:

The experiments don't include just one, but multiple runs. These runs, their parameters and their results are stored inside of a summary file.

method	volumelist	volumenum	k	max traffi	margin	ingestion	optimizati	traffic	deletion	initial vol1	final vol1	initial vol2	final vol2	initial vol3	final vol3	initial vol4	final vol4	initial vol5	final vol5
GREEDY	heavy	5	13	20	0.01	0.652803	5.16162	0.161329	0.018555	0.179132	0.195097	0.153955	0.195087	0.178636	0.195149	0.290189	0.206674	0.198088	0.20799
GREEDY	heavy	5	13	40	0.01	0.539984	10.6692	0.322681	0.040041	0.179132	0.203548	0.153955	0.193924	0.178636	0.203013	0.290189	0.199295	0.198088	0.2002
GREEDY	heavy	5	13	60	0.01	0.544439	14.1639	0.483853	0.054492	0.179132	0.201002	0.153955	0.20391	0.178636	0.20167	0.290189	0.202183	0.198088	0.19123
GREEDY	heavy	5	13	100	0.01	0.581276	22.8253	0.802319	0.074518	0.179132	0.199896	0.153955	0.210163	0.178636	0.208517	0.290189	0.196999	0.198088	0.18442

method: what algorithm was used

volumelist: the name of the file containing the volume lists. Explained in the volume list section.

volume number: number of volumes in above list

k: sample size

max traffic: input max traffic

margin: input error margin

ingestion time: how long it took to ingest

optimization time: how long it took to optimize

total traffic: how much traffic was used in total according to Greedy

total migration: how much deletion happened in total according to Greedy

volume 1 initial size: what was the initial fraction of Volume 1 in the system

volume 1 final size: what Greedy believes to be the final fraction of Volume 1 in the system

...

volume n initial size

volume n final size

8) Output:

Two outputs are created. One for debugging: the human readable file, and one to be given as input to the calculator.

The first type has the suffix of csv and is explained in the Greedy iterations section

The second type has the suffix csv_migrationPlan and is given as input to the calculator.

Important notice is that the volumes provided at the second output are the volumes **as given** to the algorithm and **not** their original equivalents. Therefore, running calc with this output would compute the value of the migration on the **sampled** system. In order to turn this output to one correlating to the original system, you will need to change the first column on which we will expand shortly.

In examples/exampleCalcInput-sampled.csv we have an example input to the calculator. We will use it to explain the file format

|Column 1 | Column 2 | Column 3 | Column 4

./example/basic_1.csv	0-1-2-3	0-1-2	1
./example/basic_2.csv	4	3-4	1
./example/basic_3.csv	5	5	1

Column1 contains the volumes presentation in the volume standard explained in the volume standard section. This volume would be used by the calculator as it simulates the migration plan to find the value of the migration plan. By changing the volumes given here to their matching original files, the file would in turn correspond into calc input for the computation of the migration plan's value on the original system.

Column2 states the file SN's in the volume at its initial state. Separated by "-"

Column3 states the file SN's in the volume at its final state. Separated by "-"

Column4 states whether the row should be considered by the calc. in the case of the automatically created outputs, this will always be 1. Providing 0 will tell the calculator to ignore it.

So, if you wish to run the **same** migration plan on a **different** system (for example the original unsampled system) you will need to **manually** change the volume locations in column1 with their equivalents.

9) Running Calc:

After creating a migration plan, we want to be able to simulate the migration plan on the system and to find how much in practice has been deleted and how much in practice has been moved.

run calc as such:

./calc -file [the migration plan in calc format] -output [where you want to save the output csv file]

For example

./calc -file exampleCalcInput-sampled.csv -output calcResult.csv

The migration plan in calc format is explained as the second type of output in the output section. The calculator would create, at the folder given under the output flag, a csv file looking like this:

workload	initial size B	initial size %	traffic B	deletion B	final size B	final size %	total traffic B	total traffic %	total deletion B	total deletion	lb score
basic_1.csv	4096	57.1429	0	1024	3072	42.8571					
basic_2.csv	1024	14.2857	1024	0	2048	28.5714					
basic_3.csv	2048	28.5714	0	0	2048	28.5714					
							1024	14.2857	0	0	1.28571

once you change the sampled volumes to the original ones(exampleCalcInput.csv) and run, you will receive the equivalent calculations for the original volumes under the migration plan.

Workload: the volume file used for the simulation

Initial size B: the initial size of the volume in Bytes

Initial size %: the initial percentage of the volume's size from the total initial size

Traffic B: the size in Bytes moved into the volume under this migration plan simulation

Deletion B: the size in Bytes deleted from the volume under this migration plan simulation

final size B: the final size of the volume in Bytes

final size %: the final percentage of the volume's size from the total final size

total traffic B: the total amount of bytes that has been moved under the simulation

total traffic %: what percentage of the **initial system size** has been moved

total deletion B: the total amount of bytes that has been deleted under the simulation

total deletion %: what percentage of the **initial system size** has been deleted

lb score: the load balancing score of the system after the migration. Calculated as the maximum volume percentage divided by the average volume percentage.

Should you change the output file to the non sampled volume version and run calc, the calc on the original result would look similar:

workload	initial size B	initial size %	traffic B	deletion B	final size B	final size %	total traffic B	total traffic %	total deletion B	total deletion %	lb score
B_heuristic_depth1_001_081_000_000.csv	1.20256E+12	35.6939	10647602997	9.94498E+11	2.18715E+11	15.8413					
B_heuristic_depth1_001_081_001_027.csv	1.17994E+12	35.0225	46712566167	7.37994E+11	4.88662E+11	35.3934					
B_heuristic_depth1_001_081_028_054.csv	9.86595E+11	29.2836	64464132330	6.81421E+11	3.69638E+11	26.7726					
B_heuristic_depth1_001_081_055_081.csv	0	0	3.03642E+11	0	3.03642E+11	21.9926					
							4.25467E+11	12.6285	1.98845E+12	59.02	1.41574

- Optional -no_cache flag to run the calc without looking up or saving results in a cache. This has short term faster runs and long term slower runs.

10) Greedy Iterations:

The greedy algorithm works using iterations. Those iterations are later turned into the migration plan. For debugging purposes, the Iterations are saved into the iteration file.

Each line presents a single iteration.

Iteration	sourceVolume	targetVolume	fileSn	replicated KB	deleted KB	moved KB	totalTraffic KB	totalMigration KB	traffic% of Total source	deletion% of Total source	Time(s)	iteration type
1	0	2	18	99.248	37.9678	99.248	99.248	37.9678	0.0603672	0.0230937	0.154878	balance
2	0	2	9	0	111.304	0	99.248	149.271	0.0603672	0.0907938	0.157244	balance
3	0	2	0	0	414.654	0	99.248	563.926	0.0603672	0.343006	0.159525	balance

Iteration: iteration number
sourceVolume: what is the index of the volume chosen as source
targetVolume: what is the index of the volume chosen as target
fileSn: what is the SN of the file chosen to move
replicated KB: how many KB were replicated
deleted KB: how many KB were deleted
moved KB: how many KB were moved
totalTraffic KB: what is the ACCUMALATIVE traffic
totalMigration KB: what is the ACCUMALATIVE deletion
traffic% of Total source: what is the ACCUMALATIVE traffic percentage from the INITIAL size
deletion% of Total source: what is the ACCUMALATIVE deletion percentage from the INITIAL size
Time(s): how much time as passed

iteration type: either balancing (moving the system from an illegal state by the error margin to a legal state by the margin) or optimizing (trying to reduce the size while remaining balanced)